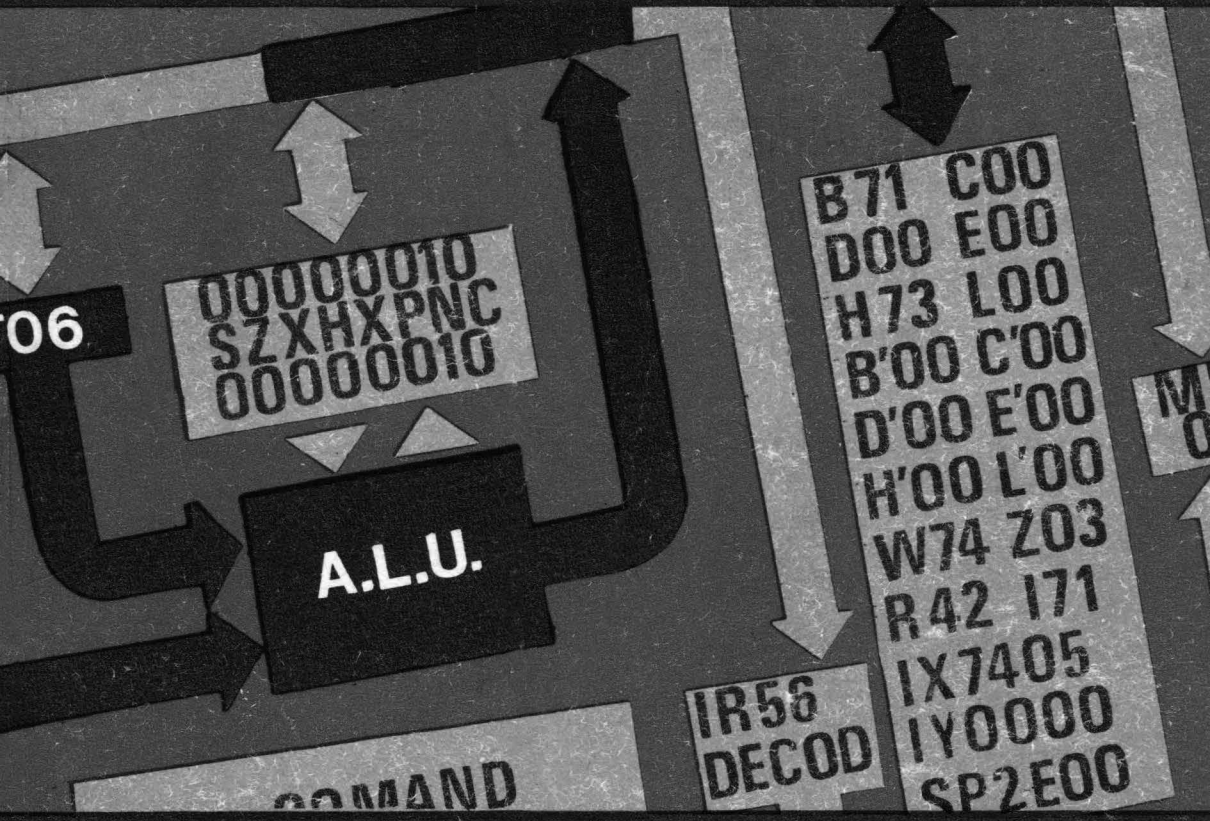
M. PATRUBÁNY

Volumul 2

TOTUL DESPRE... MICROPROCESORUL Z80



EDITURA TEHNICA

BIBLIOTECA

Automatică
Informatică

Electronică
Management

Ciclul „Totul despre ..."

Biblioteca de automatică, informatică, electronică, management. Seria PRACTICĂ

M. K. Starr. **Conducerea producției.**
T. Baron ș.a. **Calitatea produselor Manual practic.**
A. Vlădescu ș.a. **Radioreceptoare**
M. Mayer. **Tiristoare în practică. Mutatoare cu comutație forțată**
G. Moltgen. **Tiristoare în practică. Mutatoare cu comutație de la rețea**
L. Zamfirescu, I. Oprescu. **Automatizarea cuptoarelor industriale**
I. Papadache. **Automatica aplicată, ediția I și a II-a**
Șt. Alexandru. **Automatizarea proceselor tehnologice în industria lemnului**
V. H. Lisicikin. **Prognoza tehnico-științifică în ramurile industriei**
G. Raymond. **Tehnica televiziunii în culori**
T. Homoș. **Capacitatea de producție în construcții de mașini**
S. Radu, D. Filoti. **Centrale telefonice automate. Sisteme de comutație.**
R. Stere ș.a. **Tranzistoare cu efect de cîmp**
D. N. Sapiro. **Proiectarea radioreceptoarelor**
V. Antonescu, M. Popovici. **Ghid pentru controlul statistic al calității producției**
N. Stanciu ș.a. **Tehnica imaginii în cinematografie și televiziune**
P. Vezeanu, St. Pătrașcu. **Măsurarea temperaturii în tehnică**
T. Penescu, V. Petrescu. **Măsurarea presiunii în tehnică**
P. Popescu, P. Mihordea. **Măsurarea debitului în tehnică**
P. Vezeanu. **Măsurarea nivelului în tehnică**
C. Hidoș, P. Isac (coordonator,) **Studiul muncii, vol. I—VIII**
V. Baltac ș.a. **Calculatorul FELIX C-256, Structură și programare**
R. L. Morris. **Proiectarea cu circuite integrate TTL**
Ishikawa Kaoru. **Controlul de calitate pentru maiștri și șefi de echipe**
A. M. Buhtiarov ș.a. **Culegere de probleme de programare**
P. Constantinescu. **Sisteme informatice, modele ale conducerii și sistemelor conduse**
E. S. Buffa. **Conducerea modernă a producției, vol. I și II**
A. Vătășescu ș.a. Dispozitive semiconductoare. Manual de utilizare
A. Nadolo. **Măsurarea volumului și cantității lichidelor în industrie**
Ch. Jones. **Design. Metode și aplicații**
Gh. Pisău ș.a. **Elaborarea și introducerea sistemelor informatice**
C. Hidoș. **Analiza și proiectarea circuitelor informaționale în unitățile economice**
A. Vătășescu ș.a. Circuite integrate liniare. Manual de utilizare vol. 1, 2, 3, 4
M. Silișteanu ș.a. **Scheme de televizoare, magnetofoane, vol. 1 ș, 2 ed. a II-a**
D. W. Davies. **Rețele de interconectarea calculatoarelor**
V. Pescaru ș.a. **Fișiere, baze și bănci de date**
Gh. Baștiurea ș.a. **Comanda numerică a mașinilor-unelte**
N. Sprînceană ș.a. **Automatizări discrete în industrie. Culegere de probleme**
M. Florescu. ș.a. **Cibernetică, automatică, informatică în industria chimică**
S. Călin. **Optimizări în automatizări industriale**
S. Maican. **Sisteme numerice cu circuite integrate**
I. Ristea ș.a. **Manualul muncitorului electronist**
M. Simionescu. **Proiectare unitară a circuitelor electronice**
C. Cluceru. **Tehnica măsurărilor în telecomunicații**
P. Nițulescu. **Electroalimentarea instalațiilor de telecomunicații**
R. Rapeanu ș.a. **Circuite integrate analogice. Catalog**
Șt. Lozneanu ș.a. **Casetofoane. Depanare. Funcționare**
T. Rădulescu ș.a. **Centrale telefonice automate**
N. Iosif ș.a. **Tiristoare și modele de putere. Catalog**
P. Postelnicu. **Sisteme și linii de transmisiuni telefonice**
M. Silișteanu ș.a. **Receptoare TV în culori**
V. Baltac ș.a. **Sisteme interactive și limbaje conversaționale**
V. Baltac ș.a. **Calculatoare electronice, grafica interactivă, prelucrarea imaginilor**
T. Geber, I. Miu ș.a. **Echipamente periferice 1, 2, 3**
A. Davidoviciu, B. Bărbat. **Limbaje de programare pentru sisteme în timp real**
M. Guran, Fl. Filip **Sisteme ierarhizate în timp real cu prelucrarea distribuită a datelor**
În ciclul Totul despre... a apărut: A. Petrescu ș.a., Totul despre... calculatorul personal a MIC

Patrubány Miklós

TOTUL DESPRE... MICROPROCESORUL Z80

Volumul 2

EDITURA TEHNICĂ
București, 1989

Prefață: ing. GH. CONSTANTINESCU
Recenzii: dr. ing. ADRIAN DAVIDOVICIU,
dr. ing. NICOLAE ȚĂPUȘ
dr. ing. LIVIU DUMITRAȘCU,

Redactor: ing. PAUL ZAMFIRESCU

Coperți: arh. SILVIA PINTEA,
arh. LIVIU DERVEȘTEANU (ilustrații)
Tehnoredactor: V. E. UNGUREANU
C.Z.: 681.3
Bun de tipar: 6 05 1989
Coli de tipar: 15

ISBN 973-31-0009-9
ISBN 973-31-0007-2

INTREPRINDEREA POLIGRAFICĂ CLUJ,
Municipiul Cluj-Napoca, cd. nr. 425/1987

Cuprins general

Volumul 1

Volumul 2

Cuprins detaliat

Volumul 2

Partea a III-a

Programarea microprocesorului Z80 : Un proiect : o casă de marcat electronică

Partea a III-a

PROGRAMAREA MICROPROCESORULUI Z80 UN PROIECT: O CASĂ DE MARCAT ELECTRONICĂ

10
INTERREGNUM

"Dacă zidarii ar construi casele în aceeași manieră în care programatorii își creează programele, atunci prima ciocănitoare ar distruge civilizația"

G. M. WEINBERG

Ghid de proiectare structurată.

Cei care au parcurs prima parte a acestei cărți știu totul, sau aproape totul despre structura microprocesorului și despre setul lui de instrucțiuni. *Pentru al putea utiliza eficient este necesar să se cunoască modul de programare al microprocesorului*, căci așa cum demonstrează tendințele de dezvoltare, elaborarea hardware-ului tinde să se banalizeze, concomitent cu creșterea vertiginoasă a investițiilor de efort și a cheltuielilor necesare pentru elaborarea pachetelor de software dedicate. Păstrînd măsura proporțiilor, putem afirma că una și aceeași placă de unitate centrală, echipată cu microprocesor, memorie (RAM și EPROM) și cîteva dispozitive de intrare/ieșire, poate constitui nucleul oricărui echipament, începînd cu o mașină de spălat automată, trecînd pe la calculatoarele personale pînă la roboții industriali. Elementul distinctiv va fi în toate aceste cazuri software-ul.

Dedicăm partea a doua a cărții prezentării, pe baza unui studiu de caz detailat a tehnicii de programare a microprocesorului Z80 în limbaj de asamblare, limbaj care permite exploatarea la maximum a resurselor oricărei unități centrale de calculator.

Pentru descrierea mai clară a algoritmilor, în întregul volum vom folosi și un limbaj de nivel înalt, un "pseudo" PASCAL.

Devansăm studiul de caz prin acest capitol intermediar, în care vom sintetiza cîteva principii de bază, nutrind speranța ca pe această cale să concurăm la infirmarea afirmației malițioase citată mai sus.

10.1. Software-ul : artă sau meserie ? Noțiuni de programare structurată

În scurta istorie de aproximativ 40 de ani a calculatoarelor electronice, odată cu suportul hardware au evoluat spectaculos și limbajele de programare, numărul și diversitatea lor fiind astăzi foarte mare.

Elementul care a făcut posibile afirmații de genul celui din mottoul acestui capitol, este cristalizarea relativ tîrzie a unor tehnologii pentru elaborarea pachetelor de software. Recomandările cu caracter general, universal acceptate, s-au impus abia la mijlocul deceniului trecut, adică după aproape 30 de ani de exis-

tență a calculatoarelor electronice. Fenomenul este explicabil, și se poate regăsi în orice domeniu de activitate umană : validarea unor metode de elaborare, se poate face doar după cîțiva ani buni, răstimp în care se pot consemna anomalii, erori și imperfecțiuni, imprevizibile în etapa de definire și de lansare.

În perioada de început, acceptarea sau refuzul unui pachet de software se făcea pe baza funcționalității. S-a considerat a fi program bun, acel program care rezolva corect problemele formulate în specificația sa. Odată cu trecerea anilor s-a demonstrat că este aproape exclus ca un pachet software mai voluminos să fie complet lipsit de erori. Ieșind la iveală, unele mai devreme, altele mai tîrziu-cîteodată chiar după ani de exploatare ireproșabilă, aceste erori au impus apariți, unei noi noțiuni : întreținerea (service-ul) software-ului. În aceeași direcție a acționat și necesitatea crescîndă de a adapta unele programe la condiții noi, cona cretizate prin modificarea unor elemente din specificația funcțională sau din cea a suportului hardware.

Astfel s-a ajuns la situația aparent stupefiantă ca aproximativ 80% din activitatea de software în lume să se refere la adaptarea, punerea la punct și service-ul unor programe existente, și nu la elaborarea altora noi.

În aceste condiții criteriile de calificare a programelor s-au diversificat, pe lîngă cerințele de corectă funcționare și cele de performanță (viteză de execuție, capacitate), apărînd cele legate de ușurința cu care programul considerat poate fi înțeles de alții și cu ușurința cu care el va putea fi modificat pentru o nouă implementare.

La urma urmei aceste cerințe de calitate au impus *respectarea unor canoane* (reguli) de elaborare, care luate în ansamblu formează *tehnologia programării*.

Un set din aceste recomandări se constituie în a forma metodologia programării structurate, una din tehnicile cele mai eficiente.

Programarea structurată își propune să elaboreze produse software în care să se distingă clar structurile principale ale programului (aidoma structurilor de rezistență a clădirilor), structuri care vor fi *proiectate*, *programate* și *testate* înainte de a aborda orice problemă de detaliu.

Stilul acesta de abordare a problemelor, începînd cu ansamblul și coborînd treptat la detalii (top-down) caracterizează fiecare etapă de lucru pe parcursul elaborării unui produs program structurat.

În lucrarea lui, S. Williams [14] sintetizează principalele recomandări de programare formulate în lucrările [12]—[18], [20], [23].

Vom adopta aceste recomandări pentru microprocesorul Z80, completîndu-le cu altele rezultate pe baza experienței noastre.

Recomandări generale

a) *Proiectați programul înainte de a începe să-l scrieți.* Programe bine proiectate din punct de vedere logic se scriu și se pun la punct mult mai ușor decît cele a căror scriere s-a demarat fără rumegarea consistentă a problemei. Cu cît se consumă mai mult timp pentru proiectarea unui program, cu atît mai ușor se va realiza acel program.

b) *Proiectați programe structurate.* Programele structurate se pun mult mai ușor la punct și se pot modifica mai ieftin (adapta la noi cerințe) decît cele nestructurate (de exemplu : programele „cîrnați").

c) *Începeți totdeauna proiectarea cu nivelul ierarhic superior*, abordînd problema din punctul de vedere al utilizatorului.

d) *Impuneți-vă convenții de programare*, și folosiți-le cu maximă perseverență în întregul program.

e) *Includeți testarea modulelor în procesul de elaborare*, avansînd spre abordarea detaliilor cu nivele ierarhic superioare puse la punct.

f) *Comentați cît mai bine programele*, astfel încît ele să poată fi înțelese și de alții.

g) *Căutați un partener care să revizuiască tot ceea ce ați făcut.*

h) *Acordați nume cît mai sugestive modulelor create.* Insistînd asupra acestei activități, un „naș" bun poate scuti pe el însuși și pe colegi de multe linii de comentariu explicativ.

Vom parcurge în continuare principalele etape de lucru : *proiectarea, elaborarea, testarea, finalizarea.*

Proiectarea programelor (de sus în jos, de la ansamblu la detaliu)

Metoda sugerată este contrară instinctului. Se cere ca să proiectăm module software care nu fac aproape nimic, neștiind încă cum se va rezolva oricare problemă concretă. Totuși aceasta este calea cea bună.

a) *Puneți pe hîrtie descrierea* a ceea ce urmărește programul. În cazul pachetelor mai complexe, *scrieți documentația de utilizare a acelui program înainte de a fi scris nici măcar o linie din program.* Numai astfel, încercînd a scrie, vă puteți pune în situația utilizatorului. Pe parcursul elaborării manualelor de utilizare va trebui să clarificați multe aspecte care v-au scăpat la specificarea produsului sau pe parcursul elaborării proiectului logic.

b) *Proiectați structurile de date înainte de a scrie nici măcar o linie din program.* Structurile de date constituie un element esențial al proiectului logic, ele putînd juca un rol determinant în tehnicile de programare pe care va trebui să le folosiți.

c) Împărțiți problema în *module funcționale* și elaborați descrierea lor într-un *limbaj descriptiv* sau pe *organigrame.*

d) Proiectați *programele de interfață* dintre modulele definite, specificînd clar *fluxul informațional intermodule.*

e) *Specificați modul în care veți testa* fiecare modul elaborat. Dacă din start nu puteți întrezări o metodă de testare cît mai eficientă, restructurați de pe acum modulele.

Implementarea și testarea modulelor

Și această activitate se va desfășura de sus în jos, în paralel cu elaborarea modulelor program.

a) *După ce ați elaborat un modul program testați-l și puneți-l la punct.* Veți întreba cum anume se poate pune la punct un program care apelează rutine care încă nici nu sînt concepute ? Aceste rutine le veți înlocui cu **rutine „oarbe"**, care nu fac nimic (RET) sau, returnează o valoare constantă, sau vă imprimă un mesaj de genul : **„ai fost la mine : x"**, unde **x** este numele rutinei **„oarbe"**.

apelate. (În literatura de limbă engleză aceste module se numesc destul de spiritual și **„stubroutine"**, „praf" (în ochi ?), în loc de **„subroutine"**. Testînd astfel toate ramurile modulului elaborat, puteți avansa în munca de elaborare a programului, substituind treptat rutinele „oarbe" cu cele reale.

b) *Fiți pregătiți să modificați (eventual să reproiectați) unele module ierarhic superioare, în măsura în care avansați la cele inferioare, dacă rutinele reale impun acest lucru.* Veți scăpa astfel de munca destul de anevoioasă de rescriere a programelor de interfață dintre module, rescriere a cărei necesitate apare de obicei la detectarea unor anomalii pe parcursul testării *finale*.

c) După terminarea unor module mai mari, *rugați pe cineva să vă revizuiască programele.* Această etapă de lucru este deosebit de importantă, din mai multe motive :

- Există probabilitatea reală ca cel care a elaborat un program să cadă mereu în aceeași „capcană logică", scăpîndu-i astfel de repetate ori o eroare. Modul de gîndire, iminent diferit, al unei alte persoane va permite detectarea facilă a unor astfel de sincope.
- Pe parcursul acestei colaborări ambii parteneri pot învăța unii de la alții.
- Pe această cale primiți un *„feedback"* despre inteligibilitatea programului dvs.
- Încercînd să explicați lectorului unele secvențe din program s-ar putea să descoperiți chiar dvs. unele erori care pînă în acel moment v-au scăpat.

Urmarea acestor recomandări este desigur dificilă pentru începătorul absolut. El va trebui să se familiarizeze mai întîi cu limbajul pe care intenționează să-l folosească, să-și formeze anumite deprinderi, studiind și modificînd programe existente, iar doar după aceea să-și propună elaborarea unui program nou, caz în care îi recomandăm să urmeze recomandările formulate mai sus.

În continuare vom înșira cîteva intimități de programare.

Recomandarea unor tehnici de detaliu

■ *Nu folosiți niciodată coduri de instrucțiune ca date.* Nu modificați codur de instrucțiune prin program. Urmărirea, punerea la punct și înțelegerea unor astfel de programe este deosebit de anevoioasă.

■ *Încercați să cadrați fiecare modul program pe o pagină.* Dacă el se va dovedi a fi mai lung, atunci transformați părți din el în subrutine, astfel încît să realizați prezenta cerință.

■ *Încercați să dirijați instrucțiunile de salt astfel încît destinația lor să fie mai jos pe pagină.* Astfel asigurați citirea programului după modul obișnuit de citire al unui text. (Recomandarea nu este valabilă pentru salturile de la sfîrșitul buclelor, care se vor efectua obligatoriu în sus).

■ *La folosirea unor instrucțiuni de salt condiționat, încercați să le aranjați atsfel încît lista programului să continue cu condiția falsă.* Condiția adevărată ar trebui să fie (pe cît posibil) locată într-un modul separat. Astfel permitem utilizatorului să identifice mai ușor bucla principală a modulului respectiv. În caz contrar s-ar putea să-i obligați pe urmaș să răsfoiască pagini întregi de listing pentru a identifica cele 10 instrucțiuni ale buclei principale.

Exemplu : Dacă vă propuneți ca într-un program să se lanseze activități diferite pentru cazul în care tastați literele A, B, C, D, bucla principală ar trebui să se constituie după cum urmează :

```
LOOP : CALL INKEY        ; se citește tasta
       CP   "A"
       JP   Z,JOBA
       CP   „B"
       JP   Z,JOBB
       CP   „C"
       JP   Z,JOBC
       CP   „D"
       JP   Z,JOBD
       JP   LOOP
```

■ *Incercați să elaborați module cu un singur punct de intrare și un singur punct de ieșire.* Pe cît posibil, *intrarea* să fie *prima* instrucțiune de pe pagină, iar *ieșirea ultima.*

■ *Evitați salturile care trec de limita paginii respective.* În mod normal recomandarea nu se referă la salturile care se efectuează la module separate și din care nu se mai revine în pagina curentă. (de exemplu : rutină comună de tratare a unor erori.)

■ *Folosiți cît mai multe nume simbolice pentru datele care la o nouă implementare s-ar putea modifica.*

■ *Folosiți nume de etichete și de simboli care să fie cît mai sugestive,* permițînd identificarea funcției lor deja pe baza numelui simbolic.

■ *Folosiți totdeauna etichete pentru adresele de destinație a unor salturi.* Evitați salturile referite printr-un *deplasament* față de *valoarea curentă a contorului program* al asamblorului. În acest din urmă caz, inserarea sau eliminarea oricărei instrucțiuni din corpul programului va da peste cap adresa de destinație a saltului.

■ *La intrarea într-o subrutină verificați încadrarea parametrilor în limitele de valori pe care le admiteți.* În acest caz nu se vor întîmpla evenimente necontrolate la apariția unor valori de apel neprevăzute. Cea mai bună soluție este cea de a genera în aceste cazuri, un mesaj de eroare, care să identifice locul anomaliei și valoarea neprevăzută primită.

■ *Limitați pe cît posibil comunicația între subrutine la un registru, mereu același.* Nu vă așteptați niciodată ca la revenirea din subrutină valoarea acelui registru să fie nemodificată.

■ *Salvați și restaurați în subrutine toți regiștri, cu excepția celui desemnat pentru comunicație.* Vă scutiți astfel de multă bătaie de cap la implementarea programului, cînd (nerespectînd această recomandare) o întrebare plină de nedumerire *„Cine mi-a stricat registrul B?"* va fi destul de frecventă.

■ *Evitați să intercalați instrucțiuni de* **salt condiționat** *între două operații de stivă,* **PUSH** *și* **POP.** Dacă pe o ramură veți uita să reechilibrați stiva, prin efectuarea aceluiași număr de **POP**-uri și **PUSH**-urile parcurse în amonte, instrucțiunea de revenire din subrutină **(RET)** nu va încărca valoarea adresei de revenire în **PC,** ci o dată oarecare, caz în care programul *„o va lua în bălării".*

■ *Mai ales în faza de început a carierei de programator, evitați pe cît posibil folosirea instrucțiunilor* **EX (SP),HL ; EX (SP),IX** sau **EX (SP),IY.**

● *Nu folosiți instrucțiunea* **EXX** *pentru salvări curente de regiștri.* Urmărind listingul vă va fi greu să „știți" în fiecare moment, care din setul de regiștri este cel activ. Recomandăm să utilizați regiștri secundari pentru programarea unor variabile globale, și/sau ca set alternativ de lucru în rutinele de tratare a întreruperilor. Veți putea folosi în schimb această instrucțiune **(EXX)**, ori de cîte ori cerințele de viteză nu permit salvări pe stivă sau în memoria de lucru.

Dacă veți fi reușit să puneți la punct programul astfel încît el să funcționeze aparent fără erori, să nu credeți că treaba a fost terminată. Greul de-abia începe : trebuie să puneți la punct documentația de implementare și să o actualizați pe cea de utilizare.

Documentarea programelor

Documentația de utilizare a unui program este de obicei un material distinct, destinat utilizatorilor produsului finit, pentru acei, pe care îi interesează modul de operare al programului și nu modul în care el rezolvă problemele. Cînd definitivați această documentație, e bine să încercați să vă transpuneți în „pielea" utilizatorului, pentru a-i putea fi realmente de folos.

Documentația de implementare este la fel de importantă ca și cea de utilizare. Calitatea documentației de implementare poate juca un rol determinant în viața unui produs software. Dacă ea nu este inteligibilă sau dacă are lipsuri, atunci efortul de asimilare va fi probabil prea mare, fapt care va determina pe noul implementator să ia totul de la început, rescriind întregul program.

Pentru a nu se putea pierde, recomandăm includerea documentației de implementare sub forma unor comentarii în însăși corpul programului.

Iată sugestiile noastre.

a) *Includeți la începutul programului o descriere a funcției principale a programului.* Treceți în revistă principalele module ale programului, precum și jonctiunea lor. Amintiți principalele convenții folosite, tot aici.

b) *Includeți și instrucțiunile (comenzile) necesare pentru o nouă generare a codului din programul sursă.*

Aceasta este destinația originală a fișierelor apelate prin comanda **SUBMIT**, (CP/M, ISIS II, SFDX), care vor include toate comenzile necesare pentru constituirea codului obiect, care poate proveni din mai multe fișiere sursă. Într-un caz ideal, fiecărui program sursă ar trebui să i se atașeze și un fișier de generare cod, apelabil prin comanda **SUBMIT**.

c) *Includeți o descriere clară a fiecărei structuri de date importante, la* începutul blocului de date sau a modulelor care vehiculează aceste date.

d) *Prevedeți la începutul fiecărei rutine o descriere textuală a funcției rutinei respective, a regiștrilor afectați și a celor folosiți pentru transferul informațional.*

e) *Specificați în clar funcția fiecărei secvențe de cod.* Dacă undeva folosiți artificii sau manevre mai greu inteligibile, descrieți detailat tehnica folosită.

f) *Structurați programul prin folosirea spațiilor și a liniilor goale,* astfel încît entitățile distincte că *„sară în ochi".*

g) *Comentați* pe cît se poate fiecare linie a programului sursă, mai ales *secvențele greu inteligibile.*

Dacă ați parcurs cu atenție recomandările făcute în prezentul paragraf, veți putea da singuri răspunsul la întrebarea formulată în titlu.

Este incontestabil faptul că activitatea de programare este una pur intelectuală, abstractă. Este normal ca pe parcursul acestei activități să cădem în ispita unor aventuri spirituale, complexe. Spiritul inovator este un alt imbold care ne împinge spre adoptarea unor soluții nemaiîntîlnite. Iată de ce programarea poate fi considerată **artă.**

Dar odată cu răspîndirea pe scară largă a calculatoarelor și în contextul cerințelor impuse modulelor software, enunțate la începutul paragrafului, va trebui adesea să acceptăm compromisul simplității.

Nu se poate concepe o industrie de software, fără respectarea regulilor de „conviețuire" amintite. Iată de ce programarea coboară treptat din sferele înalte, devenind pe zi ce trece tot mai mult o **meserie,** *cu reguli care trebuie respectate, o meserie a intelectului.*

10.2. Instrumente de lucru

Vom prezenta principalele mijloace care se vor folosi pe parcursul elaborării programelor în limbaj de asamblare.

10.2.1. Organigrama

Pentru a elabora un program care să rezolve o problemă dată, este necesar ca înainte de toate să extragem *esența problemei*, spărgînd soluția în pași individuali de efectuat. Secvența activităților astfel obținute se numește **algoritm.** Ca să ilustrăm acest procedeu vom algoritmiza procedura de realizare a unei legături telefonice.

1. Ridicăm receptorul și așteptăm tonul.

2. Dacă tonul nu sosește în cîteva secunde, închidem receptorul și reluăm activitatea începînd cu punctul 1.

3. Dacă tonul a sosit, formăm numărul dorit și așteptăm formarea apelului.

4. Dacă sosește semnalul de „ocupat", atunci închidem receptorul și reluăm activitatea începînd cu punctul 1.

5. Dacă se formează apelul, așteptăm ca apelatul să răspundă.

6. Dacă apelatul nu răspunde timp de 30—40 sec., atunci abandonăm activitatea ; STOP.

7. Dacă apelatul ridică receptorul, legătura telefonică este stabilită ; STOP.

Această procedură-cunoscută de toți — are toate elementele unui program de calculator. Astfel distingem o **secvență de inițializare** (1.), trei **bucle** (2.→1. ; 4.→1. și 5.), o **ieșire anormală** (6.) și o **rezolvare dorită** a problemei (7.).

Reprezentarea grafică a unui algoritm se numește **organigramă.**

În fig 10.1. redăm organigrama algoritmului prezentat.

Organigrama se constituie dintr-o serie de căsuțe interconectate prin segmente direcționate, care indică căile de derulare a algoritmului.

Organigramele se scriu în limbaj natural și/sau folosind expresii matematice/logice

Pentru constituirea organigramelor se folosesc trei *simboluri principale* :

— *căsuța dreptunghiulară*, care specifică întreprinderea unei activități ;

— *căsuța romboidală*, care specifică luarea unei decizii ;

— *săgețile* care unesc cele două tipuri de căsuțe.

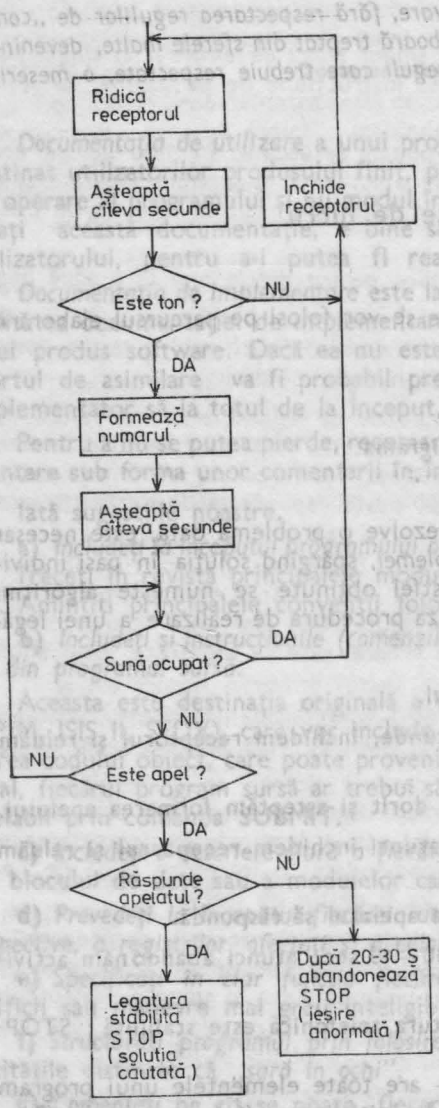


Fig. 10.1. Organigrama unui apel telefonic

Recomandăm ca *organigrama* să fie cuprinsă *pe o singură pagină*. Dacă dimensiunile ei depășesc formatul paginii, atunci organigrama se defalcă pe mai multe pagini, continuitatea lor fiind asigurată prin *căsuțe numerotate*.

Organigramele au o importanță deosebită în elaborarea programelor de calculator. Elaborarea unor *organigrame* corecte, *înainte de a începe programarea efectivă a problemei*, poate juca un rol hotărîtor în ceea ce privește timpul și efortul necesar pentru implementarea unui program. Se estimează că doar 10% din programatorii lumii știu să programeze și fără organigramă. Nenorocirea este că și ceilalți 90% consideră că fac parte din cei 10%. Drept urmare majoritatea programelor elaborate necesită eforturi mari pentru a fi puse la punct.

Cert este că odată cu creșterea experienței de programare a unui individ, organigramele se pot constitui „în cap", trecîndu-se direct la programare. Și în acest caz, lipsa unor organigrame se va resimți în procesul de service al programului, în momentul în care cineva va trebui să înțeleagă acel program.

Iată de ce vă recomandăm să folosiți totdeauna organigrama, ori de cîte ori programul dvs. depășește 10—15 linii.

În partea a doua a cărții se găsesc numeroase exemple în acest sens.

Menționăm că *organigramele nu reprezintă unicul mijloc* de fixare a unui algoritm. Datorită simplității lor, odată cu evoluția limbajelor și a conceptelor de programare, organigramele au pierdut teren pe alocuri. Au apărut *limbajele*

de nivel înalt *descriptive*, care se mulează adesea mai bine pe unele tipuri de algoritmi.

Ambele metode se vor putea folosi, ele *completîndu-se* în multe cazuri cu succes.

10.2.2. Limbajul de nivel înalt ("pseudo" PASCAL)

Limbajele de nivel înalt au apărut în anii '50, ele constituind un *salt calitativ* în dezvoltarea limbajelor de programare. Dintre numeroasele limbaje de nivel înalt create în aproximativ 30 de ani pot fi amintite : **FORTRAN, COBOL, BASIC, PL/1, PASCAL, C, ADA,** etc. *Alegerea* unuia dintre ele *depinde de problema* de rezolvat (de exemplu, **FORTRAN** se utilizează mai ales în aplicații *științifice*, **COBOL** în probleme cu *caracter economic*, **C** pentru programe de *sistem*, etc.). Utilizarea acestor limbaje a redus considerabil timpul de elaborare al programelor.

Pe lîngă *avantajele* oferite, limbajele de nivel înalt au și anumite *dezavantaje*. În primul rînd, *codul obținut* este de obicei mult mai *voluminos* decît cel obținut prin utilizarea unui limbaj de asamblare. *Avantajul portabilității* este diminuat de faptul că ele nu exploatează anumite posibilități ale hardware-ului.

Limbajele de nivel înalt (mai exact, anumite "pseudo" limbaje de nivel înalt) pot fi folosite însă și în cazul în care programul se scrie în limbaj de asamblare. În acest caz, *limbajele de nivel înalt* pot fi utilizate *pentru descrierea* mai clară a *algoritmilor*, în faza de proiectare, ele impunîndu-se mai ales *în cazul metodei top-down*.

Să luăm ca exemplu apelul telefonic din paragraful anterior. La o primă aproximare algoritmul arată astfel :

"se sună persoana dorită"
"dacă apelatul se prezintă, începe conversația"

Folosind un pseudo-**Pascal** :

```
procedure aptel
  begin
        „se sună persoana dorită"
        „dacă apelatul se prezintă, începe conversația"
  end
```

Să dezvoltăm mai departe cele două acțiuni. Cea de a doua este mai simplă :

dacă **„apelatul răspunde în 30—40 sec "**
  atunci : **„legătura stabilită"**
  altfel : **„se abandonează"**

În Pascal :

```
if „apelatul răspunde în 30—40 sec."
  then „legătura stabilită"
  else „se abandonează"
```

Prima acțiune : **„se sună persoana dorită"**, se dezvoltă mai departe în felul următor :

```
repetă
    „obține tonul"
    „formează numărul"
    „așteaptă semnalul de răspuns"
până când „sună soneria"
```

Mergînd tot așa mai departe, în final se obține următoarea procedură în pseudo — **Pascal** :

```
procedure aptel
begin
  repeat
    begin
      repeat
        begin
          if „receptorul ridicat"
          then „închide receptorul"
          else
            „ridică receptorul"
            „așteaptă cîteva secunde"
        end
      until „este ton"
      „formează numărul"
      „așteaptă semnalul de răspuns"
    end
  until „sună soneria"
  if „apelatul răspunde în 30—40 sec."
    then „legătura stabilită"
    else „se abandonează"
end aptel
```

În continuare vom prezenta pe scurt toate instrucțiunile limbajului utilizat. Prin **instr.** vom înțelege orice instrucțiune a limbajului : atribuire, apel de rutină, etc.

■ Instrucțiunea *compusă*

Un grup de instrucțiuni așezate într-un corp **begin--end**, formează o *instrucțiune compusă*, echivalentă cu o singură instrucțiune

```
begin
    instr. 1
    instr. 2
    instr. 3
end
```

■ Instrucțiunea **if**

```
if condiție
  then instr. 1
  else instr. 2
```

Dacă *condiţia* este adevărată se execută **instr. 1**, altfel se execută **instr. 2**. Organigrama corespunzătoare în fig. 10.2.

Ramura **else** poate şi să lipsească.

if condiţie
then instr.

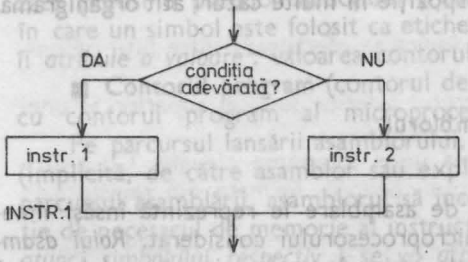


Fig. 10.2.

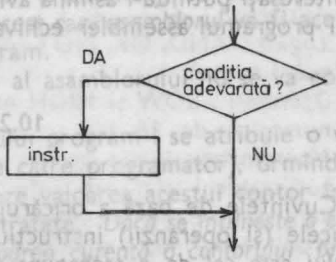


Fig. 10.3.

■ Instrucţiunea **while**

while condiţie *do* instr

Dacă *condiţia* este adevărată, se execută **instr** şi se revine la testarea condiţiei. Dacă la un moment dat *condiţia* devine falsă, se trece la executarea instrucţiunii ce urmează după **while**. Organigrama fig. 10.4.

■ Instrucţiunea **repeat**

repeat instr. *until* condiţie

instr. se execută o dată sau de mai multe ori, pînă cînd *condiţia* devine adevărată. Atunci se trece la instrucţiunea ce urmează după **repeat**. (Fig. 10.5)

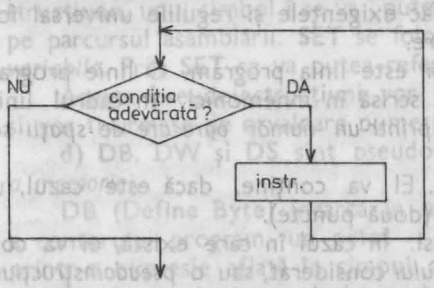


Fig. 10.4.

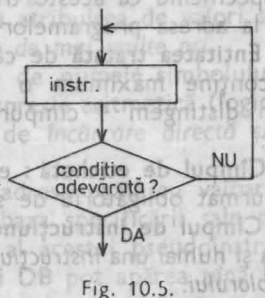


Fig. 10.5.

■ Instrucţiunea **case**

case e *of*
e1 : instr. 1
e2 : instr. 2
e3 : instr. 3
...
en : instr. n

Dacă **e = e1**, se execută **instr. 1**, dacă **e=e2**, se execută **instr. 2**

Limbajul de nivel înalt pe care vi-l propunem se aseamănă cu limbajul **Pascal**, instrucțiunile folosite reprezentînd un *subset* al acestuia.

Remarcați *desfășurarea pe orizontală* a buclelor interioare, tehnică care ușurează *identificarea părților* funcționale distincte ale programului, înlesnind *reconstituirea algoritmului.*

Pe parcursul părții a doua a acestei cărți vom folosi frecvent acest limbaj ; cei interesați putîndu-l asimila avînd la dispoziție în multe cazuri atît organigrama, cît și programul assembler echivalent.

10.2.3. Asamblorul

Cuvintele de bază a oricărui limbaj de asamblare le reprezintă însăși mnemonicele (și operanzii) instrucțiunilor microprocesorului considerat. *Rolul asamblorului este cel de a transpune programele scrise în mnemonice, în codul mașină direct executabil al procesorului.*

Un asamblor cît de cît evoluat, nu se va limita la banala traducere, ci va trebui să ofere utilizatorului facilități și servicii suplimentare. Printre acestea amintim :

- posibilitatea de utilizare a unor simboli și etichete ;
- calculul unor adrese de salt și deplasamente ;
- evaluarea unor expresii aritmetice simple, pentru ca programatorul să-și poată defini variabilele de lucru într-un mod cît mai general și flexibil ;
- posibilitatea asamblării condiționate a unor secvențe din program ;
- posibilitatea folosirii unor macroinstrucțiuni ;
- detectarea și semnalarea unor erori de sintaxă.

Vom prezenta principalele trăsături ale unui asemenea asamblor (de exemplu **M80**), specificînd că aceste trăsături satisfac exigențele și regulile universal formulate la adresa programelor de asamblare.

■ **Entitatea tratată** de către asamblor este linia program. O linie program poate conține maximum o instrucțiune scrisă în mnemonici. În cadrul liniei program distingem 4 cîmpuri, *separate printr-un număr oarecare de spații sau virgulă* :

a) Cîmpul de etichetă : este opțional. El va conține, dacă este cazul, un simbol urmat obligatoriu de semnul " :" (două puncte).

b) Cîmpul de instrucțiune : poate lipsi. În cazul în care există, el va conține una și numai una *instrucțiune a procesorului* considerat, sau o *pseudoinstrucțiune a asamblorului.*

c) Cîmpul de operanzi : va fi folosit în funcție de cerințele impuse de instrucțiunea sau pseudoinstrucțiunea care îl precede.

d) Cîmpul de comentarii : orice caracter va fi precedat obligatoriu de semnul " ;" (punct și virgulă).

Asamblorul va admite și linii vide, sau linii ce conțin doar o etichetă și/sau un comentariu.

■ **Simbolurile** : sînt un șir de 6 (sau 8) caractere, care *încep obligatoriu cu o literă*, și pot conține orice cifră și majoritatea semnelor de punctuație. (Semnele de punctuație care nu se admit în numele simbolurilor sînt specificate pentru fiecare asamblor în parte). Asambloarele acceptă *simboluri rezervate* și *simboluri utilizator.*

Setul *simbolurilor rezervate* se constituie din *mnemonicele de instrucțiuni* și *operanzi* ai microprocesorului, precum și din *pseudoinstrucțiunile* asamblorului considerat.

Simbolurile utilizator sînt definite de către programator pe parcursul elaborării programului. *Programatorul* va trebui să *atribuie valori* numerice *simbolurilor* utilizate (folosind pseudoinstrucțiuni și/sau expresii aritmetice), cu excepția cazului în care un simbol este folosit ca etichetă. În acest caz, *asamblorul* va fi acela care îi *atribuie o valoare* : valoarea contorului program.

■ **Contorul program** (contorul de locații) al asamblorului nu se va confunda cu contorul program al microprocesorului.

Pe parcursul lansării asamblorului, contorului program i se atribuie o valoare (implicită, de către asamblor sau explicită, de către programator), urmînd ca pe parcursul asamblării, asamblorul să incrementeze valoarea acestui contor în funcție de necesarul de memorie al instrucțiunilor tratate. *Dacă se întîlnește o etichetă, atunci simbolului respectiv i se va atribui valoarea curentă a contorului program.*

■ **Etichetele** definite mai sus, sînt folosite pentru a adresa zone de memorie de program (la instrucțiunile de salt), sau pe cele de date (la instrucțiunile de transfer între regiștri și memorie).

■ **Pseudoinstrucțiunile** asamblorului *nu vor fi translatate în cod executabil*, ci ele dirijează diversele activități ale asamblorului. Le vom aminti pe cele mai uzuale :

a) ORG — este o pseudoinstrucțiune care permite *inițializarea* la o valoare dorită (exprimată pe doi octeți) a *contorului de program* al asamblorului.

b) END — este pseudoinstrucțiunea de *terminare a programului sursă* și va fi locată pe ultima linie de program.

c) EQU și **SET** — sînt *instrucțiuni de atribuire* de valori pentru simbolurile utilizator. Cu **EQU** se vor defini *simbolurile constante* : folosind această pseudoinstrucțiune, unui simbol i se va putea atribui *o singură dată* o valoare numerică, pe parcursul asamblării. **SET** se folosește pentru atribuirea de valori *simbolurilor variabile*. Prin **SET** se va putea referi un simbol *de mai multe ori*.

Aceste pseudoinstrucțiuni vor fi precedate de numele simbolului selectat, și vor fi urmate de o valoare numerică sau o expresie aritmetică (logică).

d) DB, DW și **DS** sînt pseudoinstrucțiuni de *încărcare directă* și *rezervare a memoriei*.

DB (Define Byte) încarcă în memorie, la adresa egală cu valoarea curentă a contorului program, un octet — evaluat pe baza specificării sale directe sau printr-o expresie aflată în cîmpul de operanzi al acestei pseudoinstrucțiuni.

În cîmpul de operand al pseudoinstrucțiunii **DB** pot apărea pînă la 8 valori, separate prin virgulă sau spațiu, care se vor încărca la adrese de memorie succesive.

DW (Define Word) — are o semnificație asemănătoare cu **DB**, doar că ea va încărca valori numerice exprimate pe 2 octeți.

DS (Define Storage) — nu încarcă în memorie, ci incrementează valoarea curentă a contorului de program al asamblorului, cu un număr egal cu valoarea specificată în cîmpul de operand al acestei pseudoinstrucțiuni. Astfel se rezervă loc pentru cîmpuri de date ce vor fi completate pe parcursul execuției programului.

e) IF, ENDIF sînt pseudoinstrucțiuni pentru *asamblare condiționată*. Segmentul program aflat într-un corp **IF—ENDIF** va fi asamblat doar dacă condiția impusă în enunțul instrucțiunii se dovedește a fi adevărată. Condiția se impune prin

operatori relaționali **GT** (mai mare decît), **LT** (mai mic decît), **EQ** (egal) cu ajutorul cărora valoarea unui simbol se compară cu o valoare numerică sau cu o expresie.

■ **Expresiile** vor fi evaluate pe parcursul asamblării. Valoarea calculată se exprimă pe 2 octeți, transporturile de la *bit15* în sus pierzîndu-se. Într-o expresie pot apărea simboli și valori numerice. Ele vor putea fi supuse la diverse operații.

a) *Aritmetică* : **+ — * /**

b) *Logică* : **AND, OR, NOT** (operațiile logice se efectuează între biții omonimi a două cuvinte).

c) *Operatori* : **LOW** și **HIGH** extrag octetul inferior sau cel superior al unei valori numerice de 16 biți.

■ **Macroinstrucțiunile** permit elaborarea unor programe elevate. Cu ajutorul lor, utilizatorul își poate defini instrucțiuni proprii, formate din secvențe de instrucțiuni și pseudoinstrucțiuni.

Macroul se definește o singură dată, într-un corp **MACRO—ENDM**, urmînd ca numele specificat în această definiție să poată fi folosit în întregul program ca și oricare alt nume de instrucțiune sau pseudoinstrucțiune acceptată de limbajul respectiv. Pe parcursul asamblării, în locul acestei instrucțiuni „fictive", asamblorul va expanda secvența de cod din definiția macroului.

Spre deosebire de subrutine, macrourile consumă spațiu de memorie, ele expandîndu-se la fiecare apelare, dar prezintă avantajul unor posibilități de formalizare pe care subrutinele nu le oferă (parametri formali și efectivi).

■ **Instrucțiunile de control a listingului, TITLE** și **PAGE** vor imprima în fiecare pagină un titlu specificat, respectiv vor cauza salturi de pagină pentru a se asigura formatul de listing dorit.

■ **Directivele EXTERNAL** și **PUBLICS** permit referirea unor simboluri definite în alt modul program, care se asamblează separat de modulul considerat. Atribuirea valorilor pentru simbolii externi se va face de către un alt program, numit editor de legături, pe baza unei tabele de *referințe (simboli) încrucișate*, furnizată de asamblor.

Un asamblor care pune la dispoziția utilizatorului facilității de genul celor enunțate, *trebuie să parcurgă textul sursă de cel puțin 2 ori. Pe parcursul primei baleieri* se atribuie valori simbolurilor utilizate, *se constituie tabelele de simboli, macrouri și referințe* urmînd ca la a *doua trecere să se genereze efectiv codul binar*, efectuîndu-se translatarea propriu-zisă. De aceea acest asamblor va fi numit *asamblor cu doi pași (cu 2 treceri)*.

Asamblorul poate genera un listing complet, ce se poate vedea în Cap. 17, pe care-l va dirija după cerere la unul din perifericele calculatorului. (de exemplu : imprimantă, consolă sau disc). Pe listing se marchează fiecare linie eronată, de obicei în coloana întîia, indicîndu-se și tipul erorii (simbol nedefinit, simbol multiplu definit, sintaxă eronată, operand ilegal, etc.).

Codul obiect rezultat se înregistrează de obicei, direct de către asamblor, pe un suport de memorie externă (de exemplu : disc).

Prezentarea făcută este foarte sumară, dar credem că va fi suficientă pentru a putea urmări programele pe care le vom elabora în studiul de caz care urmează.

11
SPECIFICAȚIA TEHNICĂ

Ne propunem să construim împreună o casă de marcat electronică, parcurgînd toate etapele de lucru, începînd cu *specificația* constructivă și funcțională a produsului, trecînd prin definirea structurii *hardware*, implementînd apoi *programele* și terminînd cu elaborarea unor *documente scrise* care trebuie să completeze orice produs software, dacă se dorește ca acesta să poată fi înțeles și actualizat și peste cîțiva ani de zile.

Microprocesorul *Z80 va ocupa un loc central* în structura electronicii de comandă, el fiind responsabil de interfața om-mașină (tastatură, afișaj, imprimantă, sunet), de operațiile aritmetice pe care casa de marcat le are de efectuat, precum și de activitățile de gestiune și evidență a vînzărilor efectuate într-o sesiune (zi) de lucru.

Casa de marcat pe care o vom elabora este total virtuală, orice asemănare cu o casă existentă fiind o simplă coincidență.

11.1. Specificația constructivă

În continuare vom înșira principalele trăsături constructive ale casei electronice de marcat, notate cu **C.1.** — **C.8.** Împreună ele formează **specificația constructivă** (fără detalii de ordin electric și mecanic) și trebuie să stea la baza oricărui proiect, reprezentînd ghidul de lucru al proiectantului hardist. El va trebui să o respecte în întregime, într-un caz ideal. La terminarea proiectului, în raportul său final, el va consemna toate abaterile de la specificația primită.

C.1. Casa de marcat va fi dotată cu o **tastatură proprie,** care va avea următoarele taste distincte :

— cifrele zecimale de la 0—9	10 buc
— punctul zecimal	1 buc
— tastă de adunare (+)	1 buc
— tastă de înmulțire (*)	1 buc
— tastă de ștergere a ultimului număr introdus	1 buc
— tastă pentru programarea unor funcții speciale	1 buc
— tastă pentru emiterea bonului client	1 buc
Total	16 buc

C.2. *Tastatura* va avea *hardware minimal*, fiind citită de microprocesor prin program.

C.3. Casa de marcat va fi dotată cu un **dispozitiv de afişaj** constituit din diode luminiscente (LED) cu 7 segmente şi punct zecimal. Se vor reprezenta numere zecimale cu 8 cifre semnificative.

C.4. *Afişajul* va avea *hardware minimal*, fiind controlat prin baleiere de microprocesor.

C.5. Casa de marcat va fi dotată cu o **imprimantă,** care permite tipărirea pe două fîşii de hîrtie distincte a bonurilor de cumpărare. O fîşie va ieşi în exterior pentru a se putea rupe bonurile pentru client, iar cealaltă se va derula pe o rolă internă, pentru a se păstra istoria vînzărilor. Informaţia înregistrată pe cele două fîşii va fi identică.

C.6. Casa de marcat va fi dotată cu un **difuzor** pe care se vor putea emite *semnale* sonore *avertizoare*, cît mai *ergonomice*.

C.7. Casa de marcat va fi **protejată la căderile de tensiune.** În lipsa tensiunii de reţea, casa de marcat nu va funcţiona, dar va păstra *toate informaţiile* cuprinse în memoria sa *intacte*, astfel încît la reapariţia tensiunii de reţea operaţia să poată continua din punctul în care ea fusese abandonată la apariţia avariei.

C.8. Casa de marcat va fi prevăzută cu **două chei.** Cele două chei vor determina efectuarea sau neefectuarea unor operaţii. *Funcţiile de bază* (ex. : emiterea unui bon normal) se vor efectua în prezenţa primei chei (KEY0), iar cele *speciale* (ex. : anularea unui bon) se vor face doar în prezenţa ambelor chei (KEY0 şi KEY1).

11.2. Specificaţia funcţională. Nivel de detaliere 0

F.0.0. Casa de marcat va prelucra **numere** cuprinse **în gama** [0,99999999.99] Numerele pot fi numere *întregi sau zecimale*, cu 2 cifre semnificative în partea zecimală.

F.0.1. Numerele *mai mici decît 1* se vor putea introduce fie *începînd* cu *zeroul nesemnificativ* din partea întreagă, fie începînd *direct cu punctul zecimal.*

F.0.2. La *afişarea* şi/sau *imprimarea* unor numere, *zerourile nesemnificative* din partea întreagă vor fi *substituite cu blancuri* (spaţii).

F.0.3. Orice *număr introdus greşit* va putea fi *şters* prin apăsarea tastei de ştergere (CLEAR), cu condiţia ca înainte de tastarea ei să nu se fi apăsat nici o tastă de funcţie.

F.0.4. *Introducerea unui preţ* se va face cu specificarea **codului de sortiment.**

F.0.5. Distingem 100 de **clase** distincte **de mărfuri (sortimente),** codificate cu numere zecimale [0,99].

F.0.6. La introducerea preţului unui produs, **codul de sortiment** poate lipsi. În acest caz se va genera automat **codul implicit** [99].

F.0.7. *Introducerea unui preţ* se va termina prin apăsarea tastei „+".

F.0.8. La apăsarea tastei „+" *pe dispozitivul de afişaj* se va vizualiza **suma curentă** (preţul) a mărfurilor marcate pentru client, iar *pe imprimantă* se va imprima *codul de sortiment* şi *preţul introdus.*

F.0.9. *În timpul introducerii unui preţ* se va putea folosi **operaţia de înmulţire** în locul adunărilor repetate. Astfel, dacă clientul cumpără mai multe bucăţi din-

tr-un produs dat, *prețul unitar va fi înmulțit cu numărul de bucăți*, folosind tasta „*". În acest caz, la apăsarea tastei "+" se *imprimă prețul total* alocat produsului respectiv. Pe afișaj apare suma curentă a mărfurilor marcate pentru client.

F.0.10. Bonul client-normal se emite, după introducerea ultimului preț, prin apăsarea tastei TOTAL. În acest caz, *pe dispozitivul de afișaj* va apare *suma totală* a clientului, iar *pe bon se va imprima același număr* precum și *elemente de identificare* a locului și datei calendaristice de emitere a bonului.

F.0.11. Pe *fiecare bon client* se va imprima :

— **numărul unității** comerciale
— **numărul casei**
— **numărul** de identificare al **casierului**
— **numărul curent** al bonului, emis în sesiunea de lucru în curs
— **data** (ziua, luna, anul) emiterii bonului
— mesajul „**VA MULȚUMIM**".

Aceste date *se vor programa o singură dată*, la începutul sesiunii de **lucru.**

F.0.12. Pentru cazul în care *clientul nu posedă bani suficienți*, în vederea plății, se va prevedea posibilitatea de anulare a bonului introdus. *Operația de emitere* a **bonului de anulare** se va putea efectua doar *în prezența* șefului de **unitate**, prezență materializată prin existența *cheii a doua* **(KEY1)** introduse.

F.0.13. Pentru cazul în care *clientul predă* **ambalaj de schimb** (ex. : sticle) *se va* prevedea posibilitatea de a se *scădea contravaloarea acestora* din suma totală de plată. Rîndul care conține *prețul ambalajelor* restituite se va marca pe bon, el putîndu-se *distinge de celelalte rînduri de preț*, care reprezintă sume de încasat.

F.0.14. Codurile de sortiment rezervate pentru **ambalaje** vor fi primele **10.** [0,9].

F.0.15. În *memoria casei de marcat* se vor genera următoarele sume :

— **totalul vînzărilor/zi** (sesiune de lucru)
— **totalurile** vînzărilor **defalcate** pe cele 100 de sortimente/zi.

F.0.16. *Primele 10 coduri de sortiment* [0,9] *sînt rezervate pentru ambalaje* (sticle, borcane, cutii). Ele nu se vînd niciodată, ci pot fi *recuperate prin răscumpărare* de la client. De aceea numerele contorizate pentru primele 10 coduri de sortiment, vor reprezenta *bani „ieșiți" din casă*, valoarea lor *scăzîndu-se* din **totalul vînzărilor/zi.**

F.0.17. Casa de marcat va fi dotată cu **funcții speciale** care vor permite listarea, la cerere, a următoarelor date :

— **totalul vînzărilor/zi**
— **totalul** vînzărilor din cadrul unui **sortiment/zi**
— **sinteza vînzărilor,** caz în care se vor lista totalurile aferente tuturor ce.or 100 de clase de sortimente/zi.

Execuția acestor funcții speciale va fi condiționată de prezența Celei de-a 2-a chei **(KEY1).**

F.0.18. Funcțiile speciale se vor declanșa prin acționări consecutive ale tastei multifuncționale **FUNC.**

F.0.19. Casa de marcat *va fi* obligatoriu *dotată* cu **programe de test** pentru :

— memoria **RAM**
— memoria **EPROM**
— dispozitivul de afișaj

Testele se vor lansa automat în secvența de inițializare, la pornirea „rece", a casei de marcat. *Dacă* la testele **RAM** sau **EPROM** *se va detecta o eroare*, casa

de marcat *nu va deveni operațională*. Validarea **testului de afișaj** rămîne *sarcina operatorului*.

F.0.20. În caz de **operare greșită** (secvențe eronate) *casa va semnaliza* evenimentul prin desconsiderare, sau prin avertismente sonore, sau ambele. *Alegerea* uneia din cele două soluții se lasă la *libertatea implementatorului*. Recomandarea este, ca el să se ghideze după *criteriul ergonomicității*.

11.3. Funcții de bază. Nivel de detaliere 1

Cititorului atent, aidoma proiectantului, îi mai rămîn suficiente întrebări deschise privind modul detailat de funcționare al echipamentului. Vom încerca să răspundem la ele în continuare.

F.1.0. La **pornirea „rece"**, după efectuarea inițializărilor și a testelor hardware, casa de marcat va șterge dispozitivul de afișaj și înscrie pe poziția cea mai puțin semnificativă (extrema dreaptă) cifra "0" (zero).

Operatorul va tasta obligatoriu tasta **TOTAL**, care *va genera un bon vid* și va imprima *și antetul primului bon*. Astfel se va putea *testa și funcționalitatea imprimantei*, înainte de emiterea primului bon real.

Astfel se ajunge în starea de **repaus interbon**. În această stare de așteptare se poate iniția oricare din comenzile (funcțiile) casei de marcat.

F.1.1. Din starea de **repaus interbon** se pot demara următoarele acțiuni :

- **emiterea bonului client normal**
- **programarea parametrilor de stare a casei** (data, nr. casă, ...)
- **emiterea unui bon de anulare**
- **generarea totalului de vînzări pentru un sortiment dat**
- **generarea totalului general de vînzări**
- **generarea sintezei vînzărilor, defalcată pe cele 100 de sortimente.**

F.1.2. Operațiile legate de emiterea unui bon client normal le numim **funcții de bază**, iar celelalte **funcții speciale**.

F.1.3. Emiterea unui **bon client normal** *începe cu* introducerea unui *număr zecimal* de la tastatură, număr care reprezintă prețul unui produs. Plecînd din starea de **repaus interbon**, la apăsarea primei cifre, ea se va înscrie în locul zeroului inițial ("0") în poziția *cea mai puțin semnificativă* a dispozitivului de afișaj. *Următoarea* cifră tastată *va deplasa* conținutul dispozitivului de afișaj cu o poziție *la stînga*, înscriind noua cifră pe aceeași poziție, cea mai puțin semnificativă. *Dacă* se tastează *mai mult de 8* cifre consecutive, *primele cifre* tastate vor părăsi dispozitivul de afișaj prin extrema stîngă, *pierzîndu-se*. Tastarea punctului zecimal va înscrie "." în dreapta cifrei celei mai puțin semnificative. *Punctul zecimal defilează la stînga*, împreună cu cifrele introduse, la apăsarea fiecărei noi taste de cifră.

F.1.4. Pe durata **introducerii unui preț**, tastele **FUNC** și **TOTAL** sînt *desconsiderate*. Tasta **CLEAR** este *activă*. La apăsarea ei se va șterge conținutul înscris pe dispozitivul de afișaj, revenindu-se la secvența de demarare a introducerii unui preț.

F.1.5. Secvența de *introducere a prețului* se poate *termina* tastînd "*" sau "+".

F.1.6. Dacă se tastează "$*$", înseamnă că prețul introdus este un *preț unitar*, urmînd ca el să fie *înmulțit cu* numărul care urmează să fie tastat. *Înmulțitorul* se introduce după aceleași reguli ca și prețul. La *tastarea primei cifre* a înmulțitorului, dispozitivul de afișaj *se șterge*, noua cifră tastată fiind afișată.

F.1.7. Secvența de **introducere a înmulțitorului** *se termină obligatoriu cu* tasta "+". Tastele **FUNC** și **TOTAL** sînt refuzate. Tasta **CLEAR** *șterge* afișajul *și repune* casa de marcat la începutul citirii înmulțitorului.

F.1.8. La tastarea tastei "+" se **prelucrează prețul introdus.** Dacă în cursul introducerii prețului curent s-a folosit operația de înmulțire "(*)", atunci ea va fi efectuată în acest moment. *Prețul* astfel obținut este *adăugat la totalul clientului*, și *la totalul sortimentului. Suma curentă* a clientului (totalul clientului) se afișează *pe dispozitivul de afișaj. Pe imprimantă* se imprimă numărul de cod al sortimentului și *prețul recent introdus.* Dacă s-a folosit "$*$", se imprimă prețul înmulțit.

F.1.9. Astfel se ajunge în starea de **repaus interpreț.** Din starea de repaus interpreț se poate ieși tastînd un nou preț, sau tasta **TOTAL.**

F.1.10. La apăsarea tastei **TOTAL** *conținutul dispozitivului de afișaj nu se schimbă*, fiindcă el a indicat oricum totalul clientului. Pe **imprimantă** se marchează **totalul clientului,** precum și **informațiile de stare** care au fost enumerate la **F.0.11.** Formatul de imprimare se va specifica mai jos.

F.1.11. *După* prelucrarea tastei **TOTAL** *se trece* în **repausul interbon,** stare din care poate începe un nou ciclu de funcționare a casei de marcat.

F.1.12. În cazul prezentat nu s-a specificat codul de sortiment. El se generează în acest caz automat pentru valoarea 99.

F.1.13. **Codul de sortiment** se poate specifica în **repausul interbon** sau în **repausul interpreț.** Procedura este următoarea : se tastează o singură dată tasta **FUNC,** după care se tastează obligatoriu 2 cifre. Orice altă tastă, diferită de cifră, va fi desconsiderată. *Cele 2 cifre* se înscriu în pozițiile cele mai puțin semnificative ale dispozitivului de afișaj, *glisînd* de la stînga la dreapta. După prima apăsare a tastei **FUNC,** pe extrema stîngă a afișajului (cifra cea mai semnificativă), se va înscrie "1". Cele două cifre introduse reprezintă codul de sortiment. La tastarea primei cifre de cod dispare numărătorul de tastări **FUNC** ("1" în cazul de față) din poziția extremă stîngă. *Introducerea codului de sortiment se termină odată cu tastarea primei cifre din preț* : deci la tastarea celei de a 3-a cifre după **FUNC.** Această a 3-a tastă poate fi și ".". În acest *moment afișajul se șterge* și *prima cifră de preț se înscrie* în poziția cea mai puțin semnificativă. În continuare, prețul se introduce așa cum s-a prezentat mai sus.

Pe durata introducerii codului de sortiment tasta **CLEAR** este *activă.*

F.1.14. **Codul de sortiment** introdus de la tastatură, sau cel implicit (99) *se va imprima* pe rîndul prețului curent.

F.1.15. **Procedura de răscumpărare** a ambalajului de la client, respectă cu mici diferențe procedura descrisă la introducerea unui preț cu cod de sortiment *Diferențele sînt* :

— acțiunea se demarează cu două tastări succesive ale tastei **FUNC,** pornind din starea de repaus interpreț ;

— secvența nu se poate lansa în repausul interbon fiindcă ar genera sumă curentă client cu valoare negativă ;

— urmează obligatoriu codul de sortiment, cuprins obligatoriu în domeniul [00,09] ;

— la recepția tastei "+"se întreprind următoarele acțiuni :

— din totalul clientului se scade prețul ambalajelor, răscumpărate ;

— la totalul sortimentului se adună acest preț ;
— pe imprimantă se imprimă codul de sortiment, prețul introdus, urmat de semnul "—" pentru a putea fi distins de un preț direct.

După executarea acestei proceduri *se revine în repausul interpret.*

F.1.16. Dacă, pornind *din repausul interbon* numărul de tastări succesive ale tastei **FUNC** este mai mare decît *2*, atunci *se trece la* una din **funcțiile speciale.**

11.4. Funcții speciale. Nivel de detaliere 2

Casa de marcat electronică va accepta *pînă la 7 tastări succesive* ale tastei **FUNC.** Fie **n** numărul de tastări succesive. Casa va întreprinde *acțiuni diferite* pentru valori diferite ale lui **n** :

n = 1 s-a prezentat
n = 2 s-a prezentat
n = 3 **programarea parametrilor de stare**
n = 4 **emiterea unui bon de anulare**
n = 5 **generarea totalului unui sortiment dorit**
n = 6 **generarea totalului de vînzări**
n = 7 **generarea sintezei vînzărilor**

F.2.0. *Tastările succesive* ale tastei **FUNC** *se contorizează în extrema stîngă* a dispozitivului de afișaj. Tastările succesive *se termină* prin apăsarea *oricărei* taste *diferite* de **FUNC.**

La apăsarea tastei **CLEAR** se *abandonează procedura* **FUNC,** *revenindu-se în repausul interbon.* (De notat pentru operator).

F.2.1. Programarea sesiunii de lucru : n = 3.

Se va efectua *obligatoriu* următoarea secvență :

a. se introduce **numărul unității** comerciale pe 3 cifre semnificative. Dacă se introduc mai mult de 3 cifre, se vor considera ultimele 3. Tasta **CLEAR** este *activă*. Celelalte taste (**"+"**, **"*"**, **FUNC**) sînt *desconsiderate*. Introducerea parametrului *se termină* apăsînd tasta **TOTAL.**

b. se introduce fără alte tastări prealabile **numărul casei,** pe *2* cifre semnificative. Dacă se introduc mai mult de *2* cifre, se vor considera ultimele *2*. Tasta **CLEAR** este *activă*. Celelalte taste (**"+"**, **"*"**, **FUNC**) sînt *desconsiderate*. Introducerea parametrului *se termină* apăsînd tasta **TOTAL.**

c. se introduce fără alte tastări prealabile **data calendaristică** curentă, sub formă **zz.ll.aa,** unde **zz** este ziua, **ll** este luna și **aa** anul. Dacă se introduc mai multe semne atunci primele se pierd și *se consideră ultimele 8 tastări.* Tasta **CLEAR** este *activă*. Introducerea *se termină* cu tasta **TOTAL.**

d. se introduce fără alte tastări prealabile, **numărul casierului** pe 3 cifre semnificative. *Se consideră ultimele 3 tastări.* Tasta **CLEAR** este *activă*. Introducerea *se termină* prin apăsarea tastei **TOTAL.**

Astfel *se termină automat* (la cea de-a 4-a tastare **TOTAL**) *programarea sesiunii de lucru.* Parametri de stare astfel introduși se vor imprima pe fiecare bon. *Se revine în repausul interbon.*

F.2.2. Emiterea unui bon de anulare : n = 4.

Necesită prezența celei de a doua chei **(KEY1)**. Avînd bonul de anulat în față, se tastează pe rînd toate prețurile, urmate de "+". După tastarea tastei **TOTAL**, *se emite bonul de anulare* care va conține în dreptul fiecărui preț semnul **"A"**, semnalînd astfel faptul că este un bon de anulare. Dacă în bonul de anulat există *un preț de ambalaj* (marcat cu "—"), atunci el *va apare și în bonul de anulare cu marcajul* "—". Pe afișaj va apare suma totală anulată. *Suma totală* și *prețurile individuale se scad* din **totalul zilei** și din **totalurile de sortimente.**

F.2.3. Generarea totalului de sortiment : n = 5

Necesită prezența celei de-a doua chei **(KEY1)**. După **n = 5** *se tastează* 2 cifre, care reprezintă *codul sortimentului cerut*. După a doua cifră se tastează **TOTAL**. Se generează *totalul sortimentului* cerut, care *va fi afișat, și se va imprima* la imprimantă sub forma unui bon special. Formatul bonului îl vom prezenta în fig. 11.3. Pe parcursul citirii codului, tasta **CLEAR** este *activă*.

F.2.4. Generarea totalului de vînzări : n = 6

Necesită prezența celei de-a doua chei **(KEY1)**. Se tastează tasta **TOTAL** care *va imprima și afișa totalul vînzărilor* pe ziua respectivă. Formatul bonului se găsește în fig. 11.4.

F.2.5. Generarea sintezei vînzărilor : n = 7

Necesită prezența celei de-a doua chei **(KEY1)**. Se tastează tasta **TOTAL**. Ca urmare se *va genera* un **bon special** *cu 100 de rînduri. Pe fiecare rînd* se marchează *codul de sortiment* și *suma vînzărilor aferente. Afișajul este inhibat* pe durata întregii imprimări. Formatul bonului de sinteză se găsește în fig. 11.5.

F.2.6. *Dacă* în **F.2.3., F.2.4.** sau **F.2.5.** *prima tastă* după **FUNC** sau cod (la **F.2.3.**) *diferă de* **TOTAL**, atunci *operația se abandonează*, și se revine în repausul interbon.

11.5. Formatul bonurilor

Se vor emite următoarele tipuri de bonuri :

- **bon client normal**
- **bon client anulat**
- **bon cuprinzînd totalul vînzărilor pe un sortiment**
- **bon cuprinzînd totalul vînzărilor**
- **bon de sinteză,** cuprinzînd totalurile celor 100 de sortimente.

Fiecare bon va avea un **antet** în care se va specifica *numărul magazinului, al casei* și *data calendaristică* curentă. Pe fiecare bon se va imprima jos *un număr strict crescător de bon* și *codul de identificare al caseriei*. Bonurile client se vor termina cu *mesajul* **„VA MULȚUMIM".**

Formatul celor 5 tipuri de bonuri se exemplifică în fig. 11.1.—11.5.

În fig. 11.1. remarcăm faptul că fiecare preț va fi marcat cu "+", dacă codul lui este cuprins între 10—99. La restituirea ambalajelor (coduri cuprinse între 0—9) prețul se va marca cu "—". În stînga jos se găsește numărul strict crescător al bonurilor, iar în dreapta jos identificatorul casierului (018 în exemplul considerat).

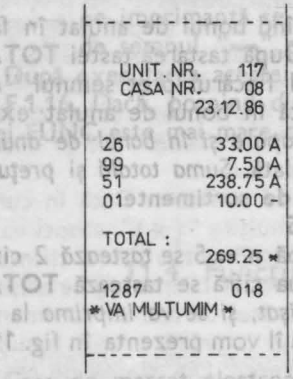


Fig. 11.1. Bonul client normal

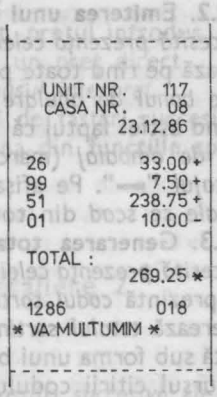


Fig. 11.2. Bonul client anulat

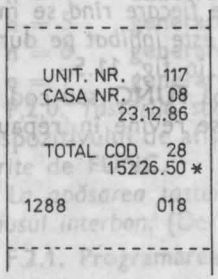


Fig. 11.3. Bon total sortiment

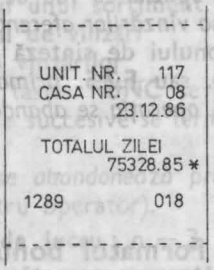


Fig. 11.4. Bon total zi

Fig. 11.5. Bon de sinteză

În fig. 11.2. se constată că fiecare preț de marfă va fi semnalat cu "A", exceptînd ambalajul restituit, care și în acest caz se scade ("—").

Bonul de sinteză din fig. 11.5., este o fîșie lungă care va conține numărul de cod de sortiment, crescînd de la 00 la 99, și vînzările aferente.

12

STRUCTURA CONSTRUCTIVĂ (HARDWARE)

În proiectarea oricărui echipament dotat cu microprocesor se disting clar *două activităţii majore : proiectarea şi elaborarea structurii fizice* **(hardware)** şi *proiectarea şi elaborarea programelor* care se înscriu în memoria nevolatilă a echipamentului, pentru a fi executate de microprocesor **(software)**. Cele două domenii diferenţiindu-se destul de substanţial şi oamenii care vor fi implicaţi în realizarea componentelor hardware şi software vor fi alţii. Pentru ca munca celor *două colective* să conveargă eficient este necesar ca, *pînă la un punct dat*, munca lor să se desfăşoare *în comun*. *Prezentul capitol* conţine această etapă de *demarare a proiectului*, etapă în care se elaborează *structura hardware-ului*. *Participarea colectivului de software* nu *este necesară* numai pentru a-şi extrage datele pentru proiectul software, ci şi pentru că în lumea microprocesoarelor chiar şi *hardware-ul este inimaginabil în absenţa unor programe de comandă*. Capitolul vrea să constituie un exemplu în acest sens.

12.1. Subansamble funcţionale

Vom prezenta în continuare soluţiile adoptate pentru realizarea *interfeţelor om-maşină conforme cu specificaţiile constructive din paragraful 11.1.*, urmînd ca *la sfîrşitul* prezentului *capitol* să putem defini *structura hardware* a casei de marcat.

12.1.1. Tastatura

Vom realiza o tastatură *controlată integral prin software*. Cele 16 taste le vom dispune *fizic* astfel încît ele să satisfacă *cerinţele de ergonomie*, iar din punct de vedere *logic* astfel încît codul care *rezultă* din interpretarea geometriei tastaturii să fie însăşi *codul pe 4 biţi a celor 16 taste*. În fig. 12.1. redăm dispunerea fizică a tastelor.

■ Grupînd tastele din punct de vedere *logic* într-o matrice de *8 coloane 2 rînduri* vom putea exploata la maximum facilităţile pe care le oferă microprocesorul **Z80**, reducînd la minimum necesarul de hardware auxiliar.

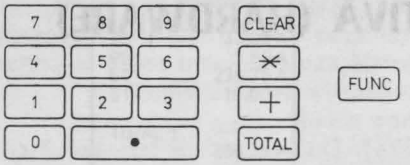


Fig. 12.1. Dispunerea fizică a tastelor

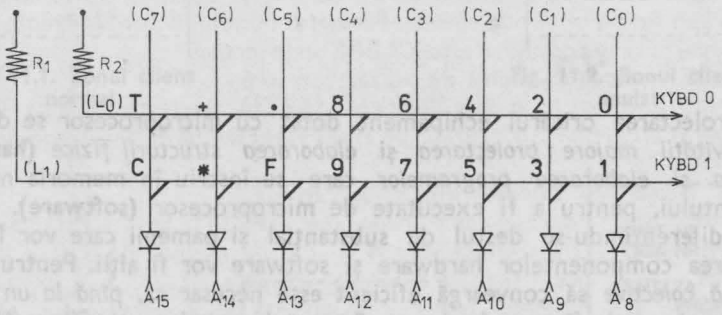


Fig. 12.2. Circuitul electric şi dispunerea logică a tastelor

■ La *intersecţia coloanelor* şi a *liniilor* se vor *dispune* tastele astfel încît la apăsarea tastei care se află la intersecţia **coloanei x** şi a **liniei y** să se *scurtcircuiteze* **coloana x** cu **linia y.**

■ Dacă nici o tastă nu este apăsată, atunci indiferent de starea coloanelor (C_0-C_7), liniile (L_0-L_7) vor fi în starea logică **"1"**, stare conferită de prezenţa celor *2* rezistenţe R_1 şi R_2.

■ *Citind* cele două linii **KYBD0** şi **KYBD1** printr-un port de intrare *microprocesorul va* putea *identifica* starea de *repaus* sau cea *activată* a tastaturii. Pentru a determina dacă oricare din cele 16 taste este apăsată, se *va genera* valoarea **"0"** pe *toate* cele 8 coloane **(A15—A8).** În acest caz *scurtcircuitul* generat între oricare linie şi coloană va *determina trecerea* în **"0"** a uneia sau a ambelor linii. Eveniment detectabil pe cale software. *Pentru a identifica* o tastă individuală, *se vor baleia* coloanele cu **"0"** (toate **"1"** cu excepţia uneia **"0"**). Constatînd care dintre cele două linii **KYBD0** şi **KYBD1** va trece în zero, se poate detecta tasta apăsată. *Numărul coloanei* se poate *codifica pe 3 bit* ($8=2^3$), iar cel al *liniei* implicate, *pe un bit*. Ataşînd cele *2* numere, *rezultă un cod pe 4 bit*, care determină univoc tasta apăsată.

■ Pentru activarea coloanelor s-au ales liniile superioare ale magistralei de adrese a microprocesorului **Z80,** datorită faptului că într-un ciclu de citire **IN,** ele conţin o informaţie ce provine dintr-unul din regiştri interni ai microprocesorului (**A** în cazul adresării directe, **B** — în cazul adresării indirecte „via" registrul **C**).

■ Diodele sînt menite să *izoleze* liniile de adresă **A8—A15,** *prevenind* astfel *scurtcircuitarea* lor, la apăsarea concomitentă a unor taste dispuse pe aceeaşi linie.

■ La elaborarea și implementarea algoritmului de citire a tastaturii va trebui să *ținem cont și de regimurile tranzitorii* (de origine mecanică) care apar *la apăsarea* și *ridicarea* unei taste (Vezi fig. 12.3).

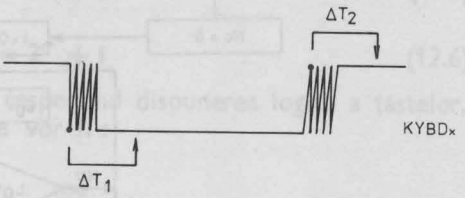


Fig. 12.3. Regimuri tranzitorii de origine mecanică la apăsarea și ridicarea unei taste (prell)

■ *Pentru a evita citirea repetată* a unei tastări (datorită vitezei mari de prelucrare a microprocesorului el s-ar putea să revină pentru citirea unei noi taste, iar regimul tranzitoriu provocat la ridicarea tastei să nu se fi terminat încă) nu este suficient a se marca prima trecere prin "**1**" a liniei detectate. Pentru a putea fi siguri de ridicarea tastei, *vom genera și o temporizare* **ΔT2** de aproximativ 0,1 **s**, a cărei valoare depinde de caracteristicile fizico-constructive ale tastaturii, și se va determina *experimental*.

Redăm în fig. 12.4. organigrama rutinei de citire a unei taste, **INKEY**.

Din organigramă se disting cîteva activități majore :

● *așteptarea apăsării ferme* a unei taste (incluzînd temporizarea △**T1** pentru a se evita regimul tranzitoriu care apare la tastare ;

● *identificarea coloanei* pe care se află tasta apăsată, prin baleierea cu un "0" singular a tuturor coloanelor și citind starea liniilor ; la ieșirea din această secvență variabila i conține numărul coloanei pe care se află tasta activată ($i \in [0,7]$) ;

● *identificarea liniei* pe care se află tasta apăsată ; se caută începînd cu prima linie (L_0) care dintre linii se activează ; incrementînd la fiecare iterație un numărător $j \in [0,1]$, la sfîrșitul secvenței el va conține numărul liniei pe care se află tasta căutată ;

● *generarea codului* pentru tasta detectată ; se atașează cele două numere i și j astfel încît ele să formeze un număr binar de 4 bit ; j se exprimă pe un bit și va fi pe poziția cea mai puțin semnificativă ;

● *așteptarea ridicării definitive* a tastei incluzînd temporizarea **ΔT2**, pentru a se evita regimul tranzitoriu.

Matematic, operația de generare a codului se poate descrie prin formula

$$Cod = 2 * i + j \tag{12.1}$$

Într-un caz general cu **C** coloane și **L** linii, numărul de biți **b**, necesari pentru a genera un cod unic, ar fi :

$$b = c + l \tag{12.2}$$

unde :

$$c = \text{int}\,(\log_2(C + \varepsilon) + 1 - \varepsilon) \tag{12.3}$$

$$l = \text{int}\,(\log_2(L + \varepsilon) + 1 - \varepsilon) \tag{12.4}$$

ε fiind un număr mic : $\varepsilon \in [10^{-3}, 10^{-6}]$

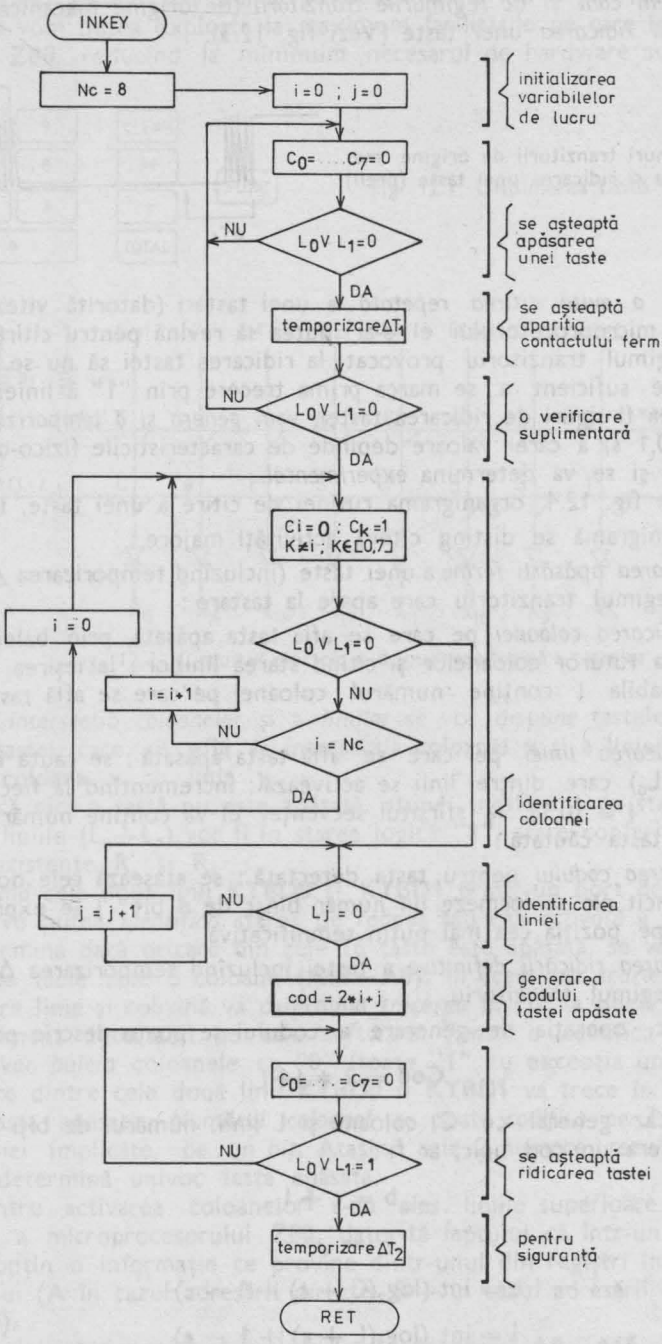


Fig. 12.4. Organigrama rutinei de citire a tastaturii (INKEY)

Codul binar se obține în acest caz aplicînd una din relațiile :

$$Cod_1 = i * 2^1 + j \qquad (12.5)$$

sau

$$Cod_2 = j * 2^C + i \qquad (12.6)$$

În cazul considerat de către noi, respectînd dispunerea logică a tastelor, indicată în fig. 12.2., codurile rezultante vor fi :

tastă	cod	tastă	cod
0	00H	8	08H
1	01H	9	09H
2	02H	•	0AH
3	03H	Func	0BH
4	04H	+	0CH
5	05H	*	0DH
6	06H	Total	0EH
7	07H	Clear	0FH

Putem trece acum la *realizarea primului program în limbaj de asamblare* : **INKEY.** Dacă se apasă concomitent pe două taste, atunci rutina va retransmite codul aceleia pe care o detectează prima dată prin succesiunea de baleiere stabilită.

Înainte de a începe elaborarea efectivă a programului stabilim următoarele :

- Rutina nu va afecta nici un registru exceptînd **A** și **F.**
- Codul tastei se va returna în registrul **A.**
- Necunoscînd încă configurația finală a portului de intrare prin care se vor citi liniile **KYBD0** și **KYBD1,** rutina va fi astfel concepută încît prin modificarea unui singur simbol ea să se poată genera pentru orice dispunere a acestor semnale la intrările portului. Condiția pe care o impunem este ca cele două semnale să fie legate la biți alăturați.

Atribuim în continuare funcții regiștrilor :

- Numărarea coloanelor **(i)** se va face în registrul **H.**
- Numărătorul de linii **(j)** va fi în registrul **L.**
- Adresa portului de intrare **SYSIN** va fi conținută în registrul **C.**
- Octetul de baleiere va fi conținut în registrul B.
- Registrul dublu **DE** se va folosi pentru a indica durata temporizărilor **ΔT1** și **ΔT2.**
- Ne propunem să alegem *nume* de etichete *sugestive* și să *structurăm programul* astfel încît algoritmul prezentat în organigramă să se poată regăsi cît mai ușor.

Iată lista programului :

```
SHIFTN  EQU   2
MASK1   EQU   0CH
        PUSH  BC          ;se salvează regiștrii
        PUSH  DE
        PUSH  HL
        LD    B,0         ;se inițializează variabilele
        LD    C,SYSIN     ;de lucru
        LD    HL,0        ;i=H, j=L
```

```
WAITDOWN: IN    A,(C)          ;se aşteaptă
          CPL                  ;apăsarea
          AND   MASK1          ;unei
          JR    Z,WAITDOWN     ;taste
          LD    DE,TDOWN       ;se aşteaptă apariţia
          CALL  TMP            ;contactului ferm
          IN    A,(C)          ;o
          CPL                  ;verificare
          AND   MASK1          ;suplimentară
          JR    Z,WAITDOWN     ;nu strică
          LD    B,0FEH

SCAN :    IN    A,(C)          ;identificarea coloanei
          CPL
          AND   MASK1
          JR    NZ,KEY
          RLC   B
          INC   H
          JR    C,SCAN
          LD    H,0
          JR    SCAN

KEY :     LD    B,SHIFTN       ;identificarea liniei
          RRCA
          DJNZ  KEY

SEARCH :  RRCA
          JR    C,FOUND
          INC   L
          JR    SEARCH

FOUND :   LD    A,H            ;determinarea codului
          RLCA                 ;tastei
          OR    L
          LD    H,A

WAITUP :  IN    A,(C)          ;se aşteaptă
          CPL                  ;ridicarea
          AND   MASK1          ;tastei
          JR    NZ,WAITUP      ;apăsate
          LD    DE,TUP         ;pentru
          CALL  TMP            ;siguranţă
          LD    A,H            ;se transferă codul
          POP   HL             ;tastei în A
          POP   DE
          POP   BC             ;regiştri restauraţi
          RET

TMP :                          ;rutină de temporizare
          DEC   DE             ;4 Tcy
          LD    A,E            ;4 Tcy
          OR    D              ;4 Tcy
          JR    NZ,TMP         ;12(7) Tcy
          RET                  ;10 Tcy
```

Urmărind etichetele regăsim principalele secvențe subliniate la descri organigramei.

- **WAITDOWN** — se așteaptă apăsarea unei taste
- **SCAN** — baleiere (identificare coloană)
- **KEY** — tastă detectată
- **SEARCH** — caută (identificare linie)
- **FOUND** — „găsit" (se generează codul tastei)
- **WAITUP** — se așteaptă ridicarea tastelor.

Fiind la primul program conceput împreună, ne permitem să *detaliăm cîteva din tehnicile folosite,*

- În secvența de așteptare a apăsării unei taste :

```
WAIT :   IN      A,(C)
         CPL
         AND     MASK1
         JR      Z,WAIT
```

Conținutul citit de pe port se inversează datorită faptului că tastele sînt active în zero. Pentru a le putea identifica concomitent, după ce toți ceilalți biți au fost eliminați, această operație de complementare este necesară *datorită faptului că* microprocesorul **Z80** *nu posedă un flag* care să indice apariția unui octet în care toți biții să fie **"1"**. În schimb *flagul* **Z (zero)** *se va înscrie* dacă toți biții din **A** devin **"0"**.

- **MASK1** este un octet care se alege astfel încît *toți biții* săi să fie **"0"** *exceptînd cei doi biți* pe care se citesc liniile tastaturii, biți care vor avea valoarea **"1"**. Astfel după execuția instrucțiunii logice **SI**, în acumulator toți biții vor fi **„0"** exceptîndu-i pe cei selectați prin **MASK1**, *care vor reda starea complementată a liniilor* **KYBD0** și **KYBD1**.

- În *secvența de baleiere* **(SCAN)** remarcăm că ea *începe* totdeauna *prin* activarea primei coloane (C_0), emițîndu-se **"0"** pe bitul cel mai puțin semnificativ :

$$\begin{matrix} & D_7 & & D_0 \\ 0FE_H = & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{matrix}$$

Activarea coloanelor se face în cel de-al treilea ciclu mașină al instrucțiunii **IN A,(C)** cînd liniile superioare $A_8 — A_{15}$ vor conține octetul de baleiere iar cele inferioare $A_7 — A_0$, vor conține adresa portului care se dorește a fi citit.

- *Baleierea efectivă* se realizează prin instrucțiunea **RLC B** ce deplasează zeroul cu o poziție la stînga. La fiecare baleiere se incrementează *numărătorul de coloană* din registrul **H (INC H)**

- *Terminarea unui ciclu de baleiere* (opt citiri urmate de rotiri) se *detectează prin* apariția valorii **"0"** *în* indicatorul de transport **Carry.** La terminarea unui ciclu de baleiere *numărătorul de coloane* se reinițializează *la* valoarea **"0" (LD H,0).**

- *După identificarea coloanei* (fapt consfințit prin efectuarea saltului **JR NZ,KEY)** cei doi biți **KYBD0** și **KYBD1** se deplasează la dreapta pînă cînd **KYBD0** *ajunge pe bitul cel mai puțin semnificativ din* **A.**

- **SHIFTN** este o variabilă care conține *numărul deplasărilor* necesare în acest sens.

- În acest moment va *începe identificarea liniei,* rotindu-se la dreapta conținutul acumulatorului *pînă cînd bitul* aferent *liniei* activate va trece *în indica-*

torul de transport. Concomitent cu fiecare deplasare se incrementează numărătorul de linie din registrul **L (INC L).**

● Dacă execuţia programului ajunge la instrucţiunea din dreptul etichetei **FOUND,** registrul **H** va conţine *numărul coloanei,* în registrul **L** regăsindu-se *numărul liniei* pe care se află tasta apăsată.

● Un programator versat va detecta imediat o modalitate de îmbunătăţire a secvenţei de identificare a liniei. Iat-o :

```
KEY    : LD    B, SHIFTN
SEARCH : RRCA
         JR C,FOUND
         INC L
         JR SEARCH
FOUND  : LD A,L
         SUB B
         RLC H
         OR    H
         LD    H,A
```

Această secvenţă presupune că octetul primit la intrarea în **KEY** are toţi biţii resetaţi, cu excepţia celor afectaţi liniilor **KYBD0** şi **KYBD1.** Numărătorul de linii va conţine în dreptul etichetei **FOUND** şi rotirile suplimentare efectuate pentru a disloca **KYBD0** pe poziţia cea mai puţin semnificativă din **A** (D_0). De aceea *numărul de rotiri suplimentare* **SHIFTN** *se va scădea* pentru a regăsi numărul liniei activate **(SUB B).** Folosind această secvenţă *se poate cîştiga timp şi spaţiu,* reducîndu-se astfel atît lungimea programului cît şi timpul de execuţie.

● Instrucţiunea **RLCA** din programul enunţat şi **RLC H** din exemplul de sus efectuează *o deplasare la stînga* a numărului de coloane, deplasare *echivalentă cu înmulţirea cu 2.*

● **Observaţie :** *Remarcăm* sintaxa instrucţiunilor de salt relativ precum şi a celei **DJNZ** în care *operandul de deplasament este* înlocuit cu *o adresă fizică,* materializată prin prezenţa unei etichete.
Exemplu :

DJNZ KEY.

În acest caz **KEY** *nu este un deplasament ci adresa fizică* a instrucţiunii la care se doreşte a se efectua saltul. *Calculul deplasamentului* respectiv rămîne *sarcina asamblorului.* Astfel se *uşurează munca programatorului,* şi se elimină o sursă ineupizabilă de erori (chiar şi cei mai experimentaţi programatori greşesc des la calculul acestor deplasamente).

● Rutina de temporizare **TMP** se bazează pe *decrementarea unui registru dublu* de la o valoare iniţială *la zero.*

Plaja de temporizare care se poate acoperi cu ajutorul acestei rutine este cuprinsă între *valorile* : (În cazul unui microprocesor funcţionînd la 2,5 **MHz,** $T_{Cy} = 400$ **ns**)

DE = 0001
$T_{min} = 29 * T_{Cy} * 1 = 11{,}6\ \mu s$

DE = 0000
$T_{max} = 65536 * 34 * T_{Cy} - 5 * T_{Cy} = 891.277{,}6\ \mu s$

● Valorile **TDOWN** și **TUP** se determină experimental datorită faptului că ele *depind de caracteristicile mecanice* ale unei tastaturi. În cazul tastaturilor cu folie a calculatoarelor **PRAE** valorile T_1=0,03 s și T_2=0,07 s conferă o *bună siguranță* de tastare, *fără să reducă* sensibil *viteza de reacție* a tastaturii.

În cap. 17 (Lista programului) la p. 1—56, se găsește lista rutinei **INKEY** care rezolvă aceeași problemă, dar este implementată în mod diferit de cazul prezentat. Pentru a nu încărca programul am eliminat temporizările de la apăsarea și ridicarea unei taste.

● *Rutina din listing* se distinge principial de cea prezentată, prin faptul că *baleierea* coloanelor *nu* începe mereu *de la coloana* **0**, *ci aleator.*

Am folosit acest prilej pentru a *exemplifica* o instrucțiune mai rar folosită **LD A,R**, care în acest caz este folosită *ca generator de numere aleatoare.* Știm deja că fiecărui număr de coloană trebuie să i se asocieze și un octet de baliere care conține un singur "**0**". În *rutina* prezentată *în listing* această asociere am rezolvat-o prin generarea unei tabele **KEYTAB** în care din doi în doi pași se regăsește de octetul baleiere și un cod de tastă aferent. Informația din **KEYTAB** *se citește o singură dată*, la apelarea rutinei după ce s-a generat numărul aleator *pentru începerea baleierii.*

Adresarea elementelor din tabelă se face adunînd la adresa de bază **(KEYTAB)** un deplasament calculat din numărul de coloană : **d = i * 2**, datorită faptului că *tabela* este *structurată pe doi octeți.*

Structura tabelei **KEYTAB** este deci :

```
KEYTAB :  octet de baleiere 0
          cod posibil 0
+2x1      octet de baleiere 1
          cod posibil 1
          .
          .
          .
+2x7      octet de baleiere 7
          cod posibil 7
```

Tehnica folosită, cea de a începe baleierea la valori aleatoare nu aduce aproape *nici un cîștig* în cazul unei tastaturi, *la* care *fenomenele* sînt *foarte lente* în raport cu viteza de lucru a procesorului. *Am prezentat-o totuși* datorită faptului că în aplicații, mai *rapide* ea ar putea fi de *folos*, precum și fiindcă ne-a permis prezentarea unor *tehnici de programare suplimentare.*

● Revenind la ideea de a *declanșa baleierea* coloanelor din *punct fix rutina* prezentată în listing *se poate scurta* sensibil, fără a se pierde din explicitatea ei.

Vom avea :

```
INKEY  : PUSH   HL
         PUSH   BC
         LD     C,SYSIN
         LD     B,0FEH
CYCLE  : LD     L,0FFH
NXTLIN : IN     A,(C)
         CPL
         AND    MASK1
         JR     Z,NOKEY
         LD     B,SHIFTN
```

```
SHIFTA  : DEC   B
          JR    Z,TSHIFT
          SRL   A
          JR    SHIFTA
TSHIFT  : CP    3
          JR    NZ,OTASTA
          DEC   A
OTASTA  : ADD   A,L
          PUSH  AF
          LD    BC,SYSIN
ASTEPT  : IN    A,(C)
          CPL
          AND   MASK1
          JR    NZ,ASTEPT
          POP   AF
          POP   BC
          POP   HL
          RET
NOKEY   : INC   L
          INC   L
          RLC   B
          JR    C, NXTLIN
          JR    CYCLE
```

Astfel s-ar reduce necesarul de memorie a rutinei de la **82** octeți la **50.**

Îl invităm pe cititor să încerce să îmbunătățească rutina prezentată, reducîndu-i și mai mult lungimea, căci se poate.

Încheiem prezentarea tastaturii sintetizînd *necesarul hardware pentru interfațarea tastaturii* :

— un șir de 8 diode (izolatoare) + 2 rezistențe
— 2 biți pe portul de intrare **SYSIN**

12.1.2. Dispozitivul de afișaj

Conform specificației tehnice (punctul **C.3.**) dispozitivul de afișaj va fi realizat cu *8 afișoare* constituite din 7 segmente și un punct zecimal (**LED—Light Emitting Diode** — diodă luminiscentă). În fig. 12.5. redăm *structura principială* a unui asemenea element de afișaj.

■ Pentru *„aprinderea"* celor 7 segmente și a punctului sînt prevăzute picioruțe. *Dacă* pe pin-ul aferent unui segment sau cel al punctului *se aplică* un semnal **TTL** de nivel **"0"** *atunci* segmentul sau punctul respectiv *va lumina.* Pentru a semnala *logica negativă*, tuturor semnalelor active în starea **"0"** le vom atașa prefixul **"N" (NSEG0,...,NDECPOINT).**

■ *Segmentele* aferente semnalelor care au *starea logică* **"1"** rămîn *„stinse"*.

■ Elementul de afișaj mai este prevăzut cu un *semnal de selecție* **NSEL,** activ în starea **"0"** care permite validarea sau inhibarea lui. *Dacă* **NSEL=0** atunci pe afișor *vor putea lumina* segmentele conform valorii instantanee a semnalelor

de intrare. Dacă **NSEL=1** atunci apare *starea inhibată* : *toate* segmentele şi punctul zecimal vor fi *stinse*, indiferent de valoarea semnalelor de intrare.

Prezenţa acestui semnal este importantă, deoarece ea *permite baleierea* mai multor elemente de afişaj alăturate. Această tehnică, numită şi *multiplexare*, reduce necesarul de hardware : nu mai este necesară ataşarea unui *element de memorare* la fiecare **LED**, ci se va *folosi unul singur* în care se va înscrie rînd pe rînd valoarea aferentă fiecărui **LED** şi se va activa **LED**-ul respectiv. Dacă această *comutaţie* se face cu o frecvenţă suficient de mare, *ochiul uman va sesiza imaginea static*, ca şi cum fiecare afişor ar fi activat în mod continuu.

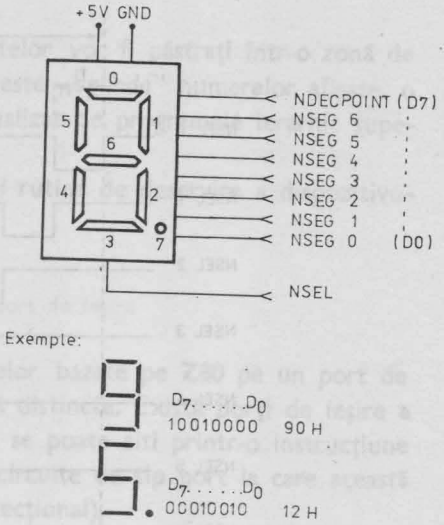


Fig. 12.5. Dioda luminiscentă cu 7 segmente şi punct zecimal

■ *Curentul mediu* care parcurge o diodă luminiscentă, şi deci şi luminozitatea ei, nu *depinde* de frecvenţa de baleiere, ci *de numărul elementelor* acţionate. Considerînd *8 elemente* afişoare, factorul de utilizare a fiecăruia este de k=1/8. Pentru a genera o intensitate luminoasă echivalentă unui regim continuu, *impulsurile de curent* injectate în starea activată a **LED**-urilor *vor trebui să fie de 8 ori mai puternice*. Acesta este *factorul care limitează* numărul de elemente care se pot ataşa într-o grupă de baleiere.

În fig. 12.6 redăm secvenţa de baleiere.

■ Pentru a minimiza hardware-ul renunţăm la folosirea unui circuit integrat decodificator, care ar decodifica numerele binare în octeţi de comandă a segmentelor. Această acţiune va fi implementată pe cale software : pe un port de ieşire (să-l numim **LEDPORT**) vom genera direct *octeţii de comandă a segmentelor*.

■ Ieşirile portului **LEDPORT** formează *magistrala de afişaj* **(NSEG0,..., NSEG6,NDECPOINT)** aşa cum se arată în fig. 12.7. Această magistrală ajunge la toate elementele de afişaj.

■ *Selecţia unui element* din cele 8 se va face acţionînd în mod succesiv afişoarele prin semnalele lor de selecţie **NSEL**. Cele 8 semnale de selecţie **NSEL0 — NSEL7** vor fi ieşirile unui circuit decodificator de tip **CDB 442 (SN 74442)**, care va decodifica un număr binar de 3 bit obţinut pe 3 linii ai portului de ieşire sistem (să-l numim **SYSOUT**).

■ *Codul binar de selecţie* **(LED2,LED1,LED0)** va fi generat de către microprocesor. Pentru a asigura o *baleiere continuă* şi uniformă, microprocesorului i se vor aplica *impulsuri de întrerupere* cu o intermitenţă de 2 ms. În cadrul deservirii fiecărei cereri de întrerupere codul de selecţie afişor va fi incrementat cu 1 şi va fi emis pe portul **SYSOUT** pentru a selecta **LED**-ul următor, iar pe **LEDPORT** se va emite octetul de activare a segmentelor **LED**-ului respectiv.

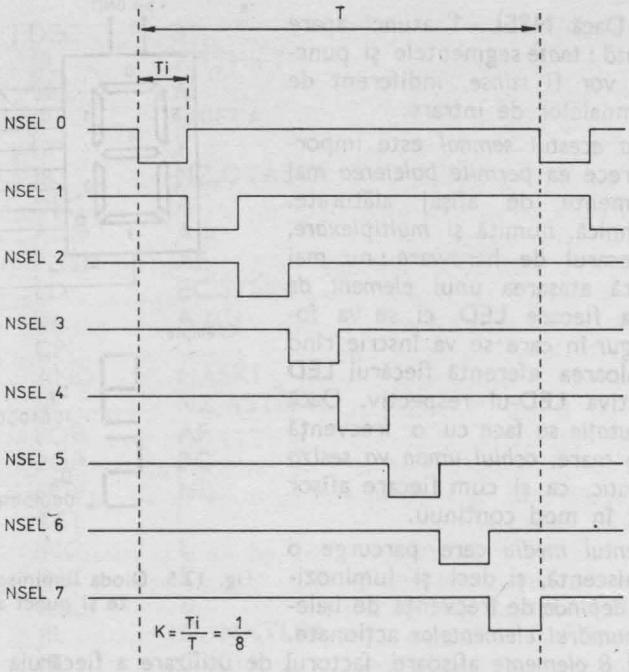


Fig. 12.6. Secvență de baleiere a unui dispozitiv de afișaj format din 8 elemente

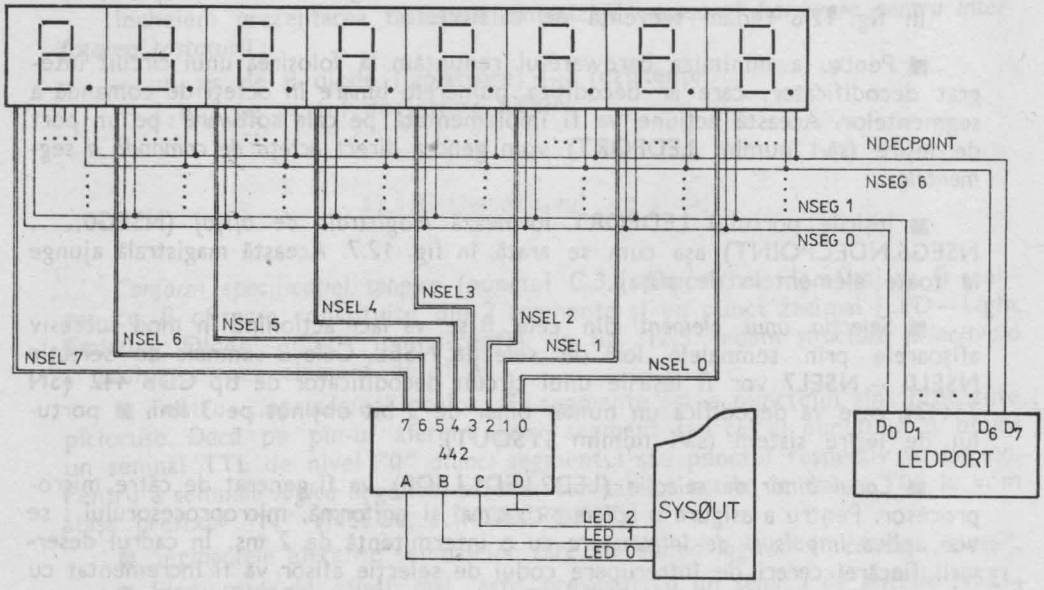


Fig. 12.7. Schema electrică de principiu a dispozitivului de afișaj

■ Cei 8 octeți de activare a segmentelor vor fi păstrați într-o zonă de memorie **RAM.** Această zonă tampon, care este *„oglinda"* numerelor afișate, o vom numi **LEDBUF,** conținutul ei fiind actualizat de programele ierarhic superioare ale casei de marcat.

Înainte de a elabora organigrama și apoi rutina de deservire a dispozitivului de afișaj introducem o noțiune nouă :

Celula martor a unui port de ieșire

În majoritatea preponderentă a proiectelor bazate pe **Z80** pe un port de ieșire se pot genera 8 semnale de comandă distincte. Există porți de ieșire a căror stare instantanee (port bidirecțional) se poate citi printr-o instrucțiune de tip **IN** (cazul circuitului **PIO**) și există alte circuite de tip port la care această manevră nu are efect (port de ieșire unidirecțional).

Ne putem imagina o structură în care anumiți biți ai unui port să aibă o funcție dedicată, iar alții să fie înzestrați cu sarcini total diferite. Acționînd unul sau un grup de semnale aferente unui subansamblu funcțional, modificarea semnalelor de comandă ale unui alt subansamblu este nedorită, uneori chiar condamnabilă.

Ținînd cont că *la efectuarea unei instrucțiuni de* tip *ieșire* **(OUT, OUTI,** etc.) *transferul afectează toți biții portului selectați,* este necesară constituirea unei celule martor în memoria **RAM,** celula a cărei conținut va fi permanent identic cu conținutul portului de ieșire tratat.

De aceea, fiecare rutină care tratează un port de ieșire unidirecțional, multifuncțional, va *citi celula martor* a portului respectiv, va modifica acei biți care în secvența respectivă prezintă interes, și va *rescrie celula martor.* Abia *după aceea* se va efectua *transferul* octetului *în portul dorit.*

În cazul proiectului de față am definit deja un port **SYSOUT.** În configurația acestuia am identificat deja 3 semnale, urmînd ca restul să-l completăm în continuare, în cadrul acestui capitol dedicat definitivării hardware-ului. Portului de ieșire **SYSOUT** îi atașăm celula martor cu numele **WITNESS.**

Celălalt port **(LEDPORT)** deservind un singur subansamblu funcțional, *nu necesită* constituirea unei *celule martor,* deoarece toate *semnalele sale* vor fi acționate concomitent și ele *deservesc un singur modul funcțional.*

În fig. 12.8 redăm organigrama rutinei de tratare a întreruperilor, adică baleierea dispozitivului de afișaj.

■ *Pentru a minimiza timpul de execuție* a acestei rutine, care într-un regim de lucru normal va fi apelată cu o frecvență de 500 **Hz,** astfel încît viteza de lucru a casei de marcat să nu fie afectată sesizabil, *vom păstra variabilele de lucru*-indicatorul de adresă din bufferul de afișaj, contorul de selecție, adresa portului de afișaj- *în regiștri interni ai microprocesorului.*

■ Ca elemente de memorare vom folosi *regiștri secundari* ai microprocesorului **Z80**, realizînd astfel un *exemplu model* privind utilizarea acestui set de regiștri.

■ Regiștrii secundari vor fi inițializați la trezirea sistemului. *Rutina* **INT** *nu va trebui să afecteze nici unul din regiștri primari.*

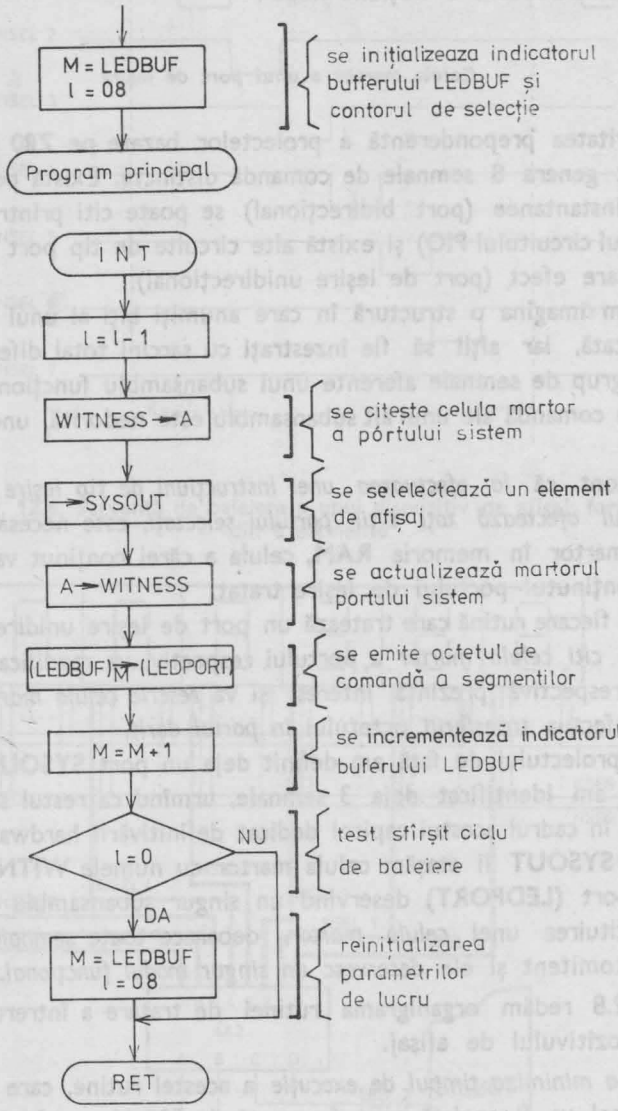


Fig. 12.8. Organigrama rutinei de deservire a întreruperilor (baleierea dispozitivului de afișaj)

Iată rutina :

```
INT :   EX      AF,AF'
        EXX
        OUTI                    ;se transferă cuvîntul de comandă
        LD      A,(WITNESS)     ;segmente
        AND     MASK2           ;poziţionarea celulei martor
        OR      B
        LD      (WITNESS),A
        OUT     (SYSOUT),A      ;se selectează LED-ul adecvat
        XOR     A
        OR      B
        JR      NZ,INT1
        LD      HL,LEDBUF       ;sfîrşitul unui ciclu de baleiere
        LD      B,BUFLEN
INT1 :  EXX                     ;se reiniţializează variabilele de
        EX      AF,AF'          ;lucru
        EI
        RETI
MASK2   EQU     0F8H
BUFLEN  EQU     8
```

Rutina se regăseşte în listingul din cap. 17 p. 1—69.

● *Reamintim* doar faptul că instrucţiunea **OUTI** transferă un octet din memorie de la adresa ind.cată de **HL** în portul de ieşire selectat prin conţinutul registrului **C**, incrementează registrul **HL** şi decrementează registrul **B**.

● Instrucţiunea **XOR A** şterge conţinutul registrului acumulator : **A** = 0.

● Modificînd variabilele **MASK1** şi **BUFLEN** rutina poate fi reasamblată pentru un dispozitiv de afişaj de orice lungime cuprinsă între [1,255] cu condiţia ca **LED**-urile să admită un curent de vîrf adecvat, care va creşte proporţional cu numărul elementelor baleiate.

12.1.3. Imprimanta

Ţinînd cont de specificaţia constructivă, punctul **C.5.**, vom alege o imprimantă de tipul **MIM—40**, care datorită simplităţii sale constructive va permite, ba chiar mai mult, va impune formarea aptitudinilor noastre în domeniul elaborării de software.

■ *Imprimanta aleasă are* un cap de scriere cu **7 ace** dispuse pe o verticală. Acele pot fi acţionate cu ajutorul a 7 electromagneţi. *Capul poate fi deplasat pe orizontală* cu ajutorul unui *motoraş* care învîrte un cilindru pe care s-a realizat un canal elicoidal. Dacă motorul se învîrte într-un singur sens, capul de imprimare va efectua o mişcare de dute/vino. Una din cele două curse se efectuează cu viteză aproape constantă, pe parcursul căreia vom efectua scrierea, acţionînd prin program acele, astfel încît ele să imprime textul dorit.

Pentru efectuarea avansului de rînd şi a returului de car nu este necesară întreprinderea nici unei activităţi de comandă. Datorită construcţiei sale mecanice, *imprimanta va efectua* la sfîrşitul cursei directe *în mod automat returul*

de car, însoţit în mod obligatoriu şi de un avans de rînd a hîrtiei de imprimare.

La extremitatea dreaptă a cursei capului se află montat un *sesizor de capăt de cursă*. La detectarea semnalului furnizat de acest sesizor se poate comanda oprirea motorului de antrenare.

Reţinem deci că mişcarea odată lansată, trebuie terminată o mişcare completă dute/vino. De aceea *imprimarea se va face rînd cu rînd* şi nu caracter cu caracter.

Cunoscînd aceste caracteristici putem elabora *schema electrică de principiu* a interfeţei de imprimantă. Ea va fi constituită din 2 blocuri de *electronică de forţă, unul* pentru comanda bobinelor *acelor*, iar *celălalt* pentru cuplarea respectiv decuplarea *motorului*. Interfaţa de imprimantă va fi deservită de un *port de ieşire* (să-l numim **PRTPORT**) şi de *un bit* al portului de intrare **SYSIN**, amintit deja la interfaţa de tastatură. Acest din urmă semnal **CARLIMIT** va permite detectarea pe cale software a capătului de cursă a capului de imprimare (Vezi fig. 12.9.).

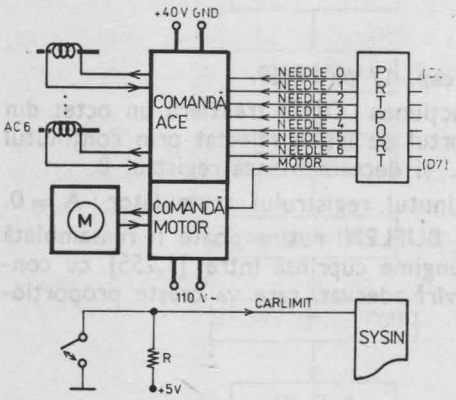


Fig. 12.9. Schema electrică de principiu a interfeţei de imprimantă

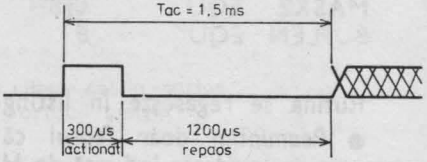
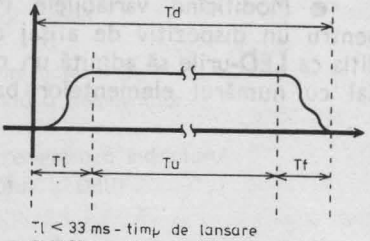


Fig. 12.10. Diagrama de timp pentru acţionarea unui ac



Tl < 33 ms - timp de lansare
Tf < 25 ms - timp de frînare
Td = 420 ms - durata cursei directe
Tu = Td - Tl - Tf = 362 ms - durata cursei utile

Fig. 12.11. Variaţia vitezei de deplasare a capului de imprimare pe durata cursei directe

Cunoscînd faptul că intervalul minim dintre 2 acţionări succesive ale unui ac este de 1,5 **ms** (vezi fig. 12.10) precum şi faptul că durata cursei utile a capului de imprimare (vezi fig. 11.11) este de 362 **ms** rezultă că vom putea imprima aproximativ 40 de caractere a 6 coloane.

$$N = \frac{T_u}{T_{ac} * N_{col}} = \frac{360}{1{,}5 * 6} = 40 \qquad (12.7)$$

■ Reamintindu-ne că imprimarea trebuie făcută pe două fîşii de hîrtie (specificaţia tehnică C.5.) lungimea oricărei linii de pe bonuri nu va putea depăşi 18 caractere. Vom putea rezerva astfel o lăţime de aproximativ 4 caractere ca spaţiu între cele două fîşii.

Putem trece atunci la elaborarea programelor ce vor deservi imprimanta.

În fig. 12.12 redăm organigrama rutinei de imprimare a unui rînd. Rutina **PRTLINE** va imprima de două ori acelaşi text, cuprins într-o zonă tampon numită **PRTBUF.**

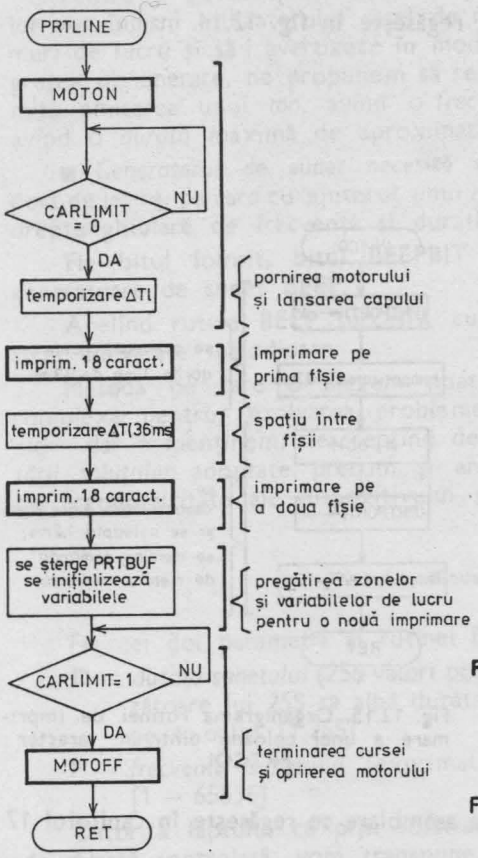


Fig. 12.12. Organigrama rutinei de imprimare a unui rînd : PRTLINE

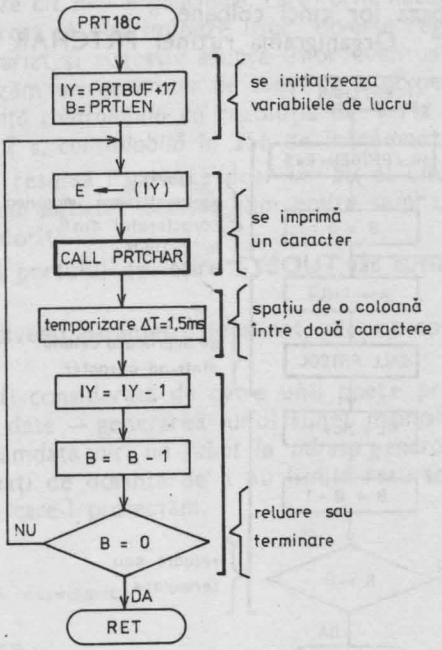


Fig. 12.13. Organigrama rutinei de imprimare a unei fîşii : PRT18C

Elaborarea unui program în limbaj de asamblare pentru implementarea unei astfel de rutine nu poate reprezenta nici o problemă. Ea se regăseşte în listingul sursă (cap. 17 p. 1—62). Amintim că între terminarea imprimării celei de-a doua fîşii şi a comenzii de oprire a motorului se poate intercala lejer *rutina de reiniţializare a bufferului de imprimare*, deoarece capul de imprimare execută între timp *cursa inversă*.

Imprimarea unei fîşii este rezolvată de rutina **PRT18C** a cărei organigramă se găseşte în fig. 12.13.

Programul aferent se regăseşte în listing (cap. 17 p. 1—63).

■ Imprimarea unui caracter se realizează cu ajutorul rutinei **PRTCHAR**, care *primește codul caracterului de imprimat în* registrul **E**. Fiecărui caracter imprimabil i se rezervă cinci octeți într-o tabelă, numită generator de caractere pentru imprimantă **(PRTGEN)**. Fiecare octet conține o mostră de biți specifică acelei coloane. *Biții* care au valoarea **"1"** *vor imprima un punct*. Știind că bitul cel mai puțin semnificativ (D_0) acționează *acul superior*, iar D_6 *acul inferior*, se poate genera conținutul *generatorului de caractere* așa cum este descris în cap. 13.

■ Descrierea caracterelor imprimabile este făcută în **PRTGEN**, în ordinea crescătoare a codurilor caracterelor. Rutina **PRTCHAR** va localiza, pe baza codului din **E**, adresa de început a secvenței de 5 octeți aferente și va imprima pe baza lor cinci coloane.

Organigrama rutinei **PRTCHAR** se regăsește în fig. 12.14.

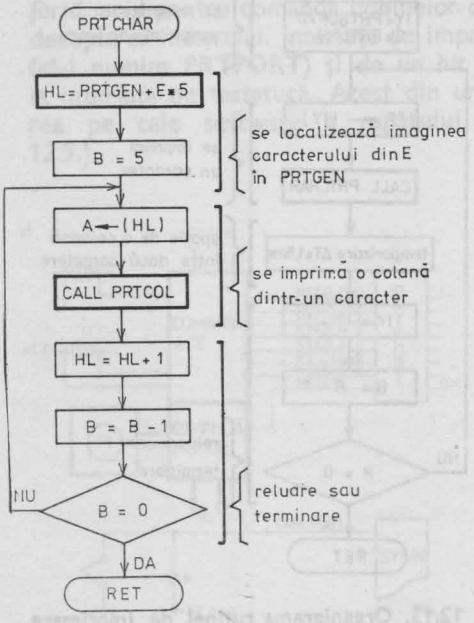


Fig. 12.14. Organigrama rutinei de imprimare a unui caracter : PRTCHAR

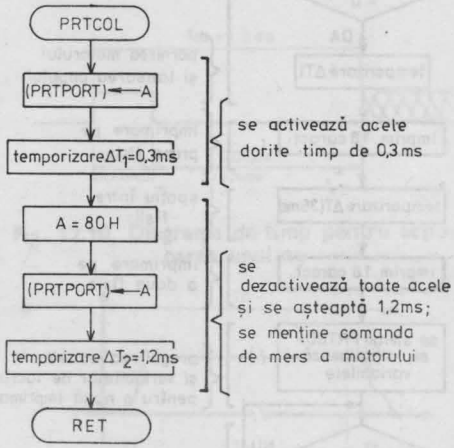


Fig. 12.15. Organigrama rutinei de imprimare a unei coloane dintr-un caracter : PRTCOL

Lista programului scris în limbaj de asamblare se regăsește în capitolul 17 pag. 1—64.

■ În sfîrșit, *imprimarea unei coloane* nu ridică nici o problemă deoarece sarcina ei este doar de a activa și a dezactiva acele dorite și de a efectua cele 2 temporizări necesare, conform diagramei de timp din fig. 12.10.

Organigrama rutinei **PRTCOL** se regăsește în fig. 12.15.

Lista programului se regăsește în cap. 17 pag. 1—65.

■ **PRTCOL** folosește ca și celelalte rutine ierarhic superioare o *rutină de temporizare de bază*, **DELAY**, care realizează temporizări egale cu un *multiplu întreg* al celei mai mici cuante de timp, semnificativă în imprimantă : **BASEDEL = 300 μs.**

■ Împreună cu rutinele de pornire și oprire a motorului de antrenare a capului de imprimare (**MOTON** și **MOTOFF**), rutina **DELAY** se regăsește în capitolul 17 pag. 1—65, 1—66.

12.1.4. Generatorul de sunet

Pentru a putea emite *semnale sonore cît mai ergonomice*, care-prin natura lor-vor înlesni utilizatorului casei de marcat să discrimineze ușor diverse regimuri de lucru și să-l avertizeze în mod variat și sugestiv asupra unor eventuale greșeli de operare, ne propunem să realizăm un *generator de sunet* care să permită emiterea unui *ton*, avînd o *frecvență controlabilă* cu rezoluția de 1 **Hz** și avînd o *durată* maximă de aproximativ *2* **s**, *controlabilă* în 256 de incremente.

■ *Generatorul de sunet necesită* ca resursă hardware doar *un bit* al unui port de ieșire, pe care cu ajutorul unui *modul software dedicat* vom emite semnale dreptunghiulare de frecvență și durată dorită.

Fie bitul folosit, bitul **BEEPBIT** al portului de ieșire **SYSOUT** iar rutina generatoare de sunet **BEEP**.

Apelînd rutina **BEEP** succesiv, cu diverși parametri de sunet, se pot construi adevărate melodioare.

Metoda pe care o alegem poate fi considerată de către unii poate prea complexă pentru rezolvarea problemei date — generarea unui sunet monofonic — dar o menținem, neacceptînd deocamdată *nici un rabat la adresa generalității soluțiilor* adoptate, precum și animați de dorința de a nu limita resursele de ergonomicitate ale dispozitivului pe care-l proiectăm.

12.1.4.1. Modelul matematic

Fie cei doi parametri ai rutinei **BEEP** :

D — *durata sunetului* (256 valori posibile, astfel ca valoarea maximă corespunzătoare lui 255 să aibă durata de aproximativ *2* **s**.) — se va reprezenta pe 1 octet [1,255]

F — *frecvența sunetului* (exprimată în **Hz** — se va reprezenta pe *2* octeți [1 — 65536]

Datorită faptului că prin software este relativ ușor a se genera impulsuri de durată controlată, vom transpune al *2*-lea parametru **F** în perioada : **T**.

Temporizarea egală cu o semiperioadă **T/2** o vom realiza folosind un ciclu de întîrziere de bază, avînd durata de **TCYCLE**.

Din considerente de implementare (modul de realizare a ciclului de întîrziere de bază ; reprezentarea aleasă pentru parametri folosiți) *ne propunem ca* **TCYCLE** *să fie* egal cu durata a *50 de tacți procesor.*

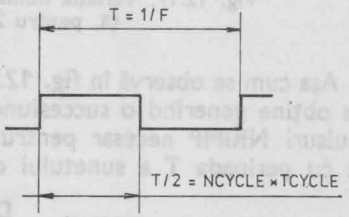


Fig. 12.16. Perioada T a semnalului de frecvență F

$$TCYCLE = 50 * TCLOCK \qquad (12.8)$$

În cazul unui microprocesor **Z80** excitat de o frecvență de tact de 2,5 **MHz**, cum este și cazul calculatorului **PRAE**, **TCYCLE=20 μs.** Această valoare permite teoretic generarea unor semnale cu frecvență limită superioare egală cu 50 **KHz**, valoare oricum neinteresantă în cazul nostru.

Numărul de repetări necesare ale ciclului de temporizare de bază **TCYCLE**, pentru a obține un semnal de frecvență **F**, va fi deci :

$$NCYCLE = \frac{T}{2 \cdot TCYCLE} = \frac{1}{2 \cdot TCYCLE \cdot F} = \frac{1}{100 \cdot TCLOCK \cdot F} \qquad (12.9)$$

Valoarea **NCYCLE** *se va recalcula la fiecare apelare a rutinei* **BEEP**. Pentru a reduce timpul de lucru al procesorului, precalculăm expresia (12.9.) : putem efectua o împărțire și o înmulțire calculînd constanta **K**.

$$K = \frac{1}{100 \cdot TCLOCK} \qquad (12.10)$$

Dorind să acordăm un *nume mai sugestiv* acestei constante, observăm că ea reprezintă întocmai numărul **NCYCLE** de temporizări de bază pentru obținerea unui semnal de frecvență 1 **Hz.**

Rezultă deci :

$$ONEHZ = \frac{1}{100 \cdot TCLOCK} \qquad (12.11)$$

În cazul calculatorului **PRAE—M** (**TCLOCK**=400 ns) **ONEHZ**=25000 iterații. Înainte de a se emite un sunet, rutina **BEEP** va calcula deci parametrul **NCYCLE** aferent.

$$NCYCLE = \frac{ONEHZ}{F} \qquad (12.12)$$

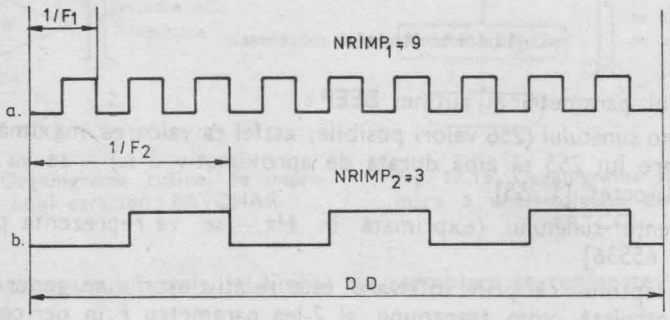


Fig. 12.17. Variația numărului de impulsuri în funcție de frecvență, pentru 2 sunete de durată identică

Așa cum se observă în fig. 12.17 durata **DD** a unui semnal sonor de frecvență **F** se obține generînd o succesiune de impulsuri avînd perioada **1/F.** Numărul de impulsuri **NRIMP** necesar pentru onbținerea duratei **DD** este invers proporțională cu perioada **T** a sunetului de frecvență F.

$$NRIMP = \frac{DD}{T} = \frac{DD}{TCYCLE \cdot NCYCLE} \qquad (12.13)$$

$K1 = \frac{DD}{TCYCLE}$ — este o variabilă proporţională cu durata sunetului dorit. O vom transforma astfel încît să calibrăm durata sunetului conform cu enunţul problemei. De aceea şi din considerente de implementare (simplitate şi volum de calcul cît mai redus) înlocuind **K1** cu

$$TIME = D \cdot 128 \qquad (12.14)$$

unde $D \in [1,255]$ este parametrul de durată folosit la apelarea rutinei **BEEP**.

Din (12.13) şi (12.14) rezultă :

$$NRIMP = \frac{TIME}{NCYCLE} \qquad (12.15)$$

Folosind expresiile (12.12), (12.14), (12.15) cei doi parametri interni, **NCYCLE** şi **NRIMP** pot fi determinaţi direct din parametri externi **F** — frecvenţa şi **D** — durata. Vom adopta aceste expresii pentru a *minimiza necesarul de rutine aritmetice*. Folosirea unei împărţiri este oricum inevitabilă pentru rezolvarea problemei, iar înmulţirea cu 128 este în aritmetica binară de-a dreptul banală.

12.1.4.2. Algoritmul

Pe baza celor expuse mai sus, algoritmul de generare a sunetului se oferă de la sine.

- Rutina **BEEP** va primi la apelare frecvenţa **F** şi durata **D** a sunetului.
- Se calculează variabilele :

TIME, NCYCLE şi NRIMP

după care se trece la emiterea sunetului.

Iată organigrama algoritmului de implementat.

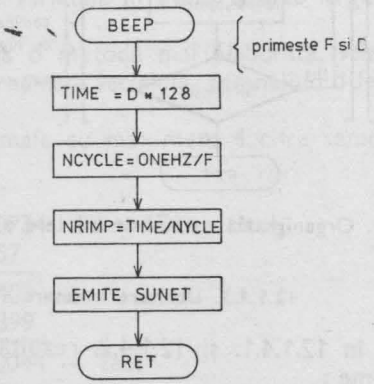


Fig. 12.18. Organigrama rutinei BEEP

Emiterea sunetului se poate concepe cum urmează :

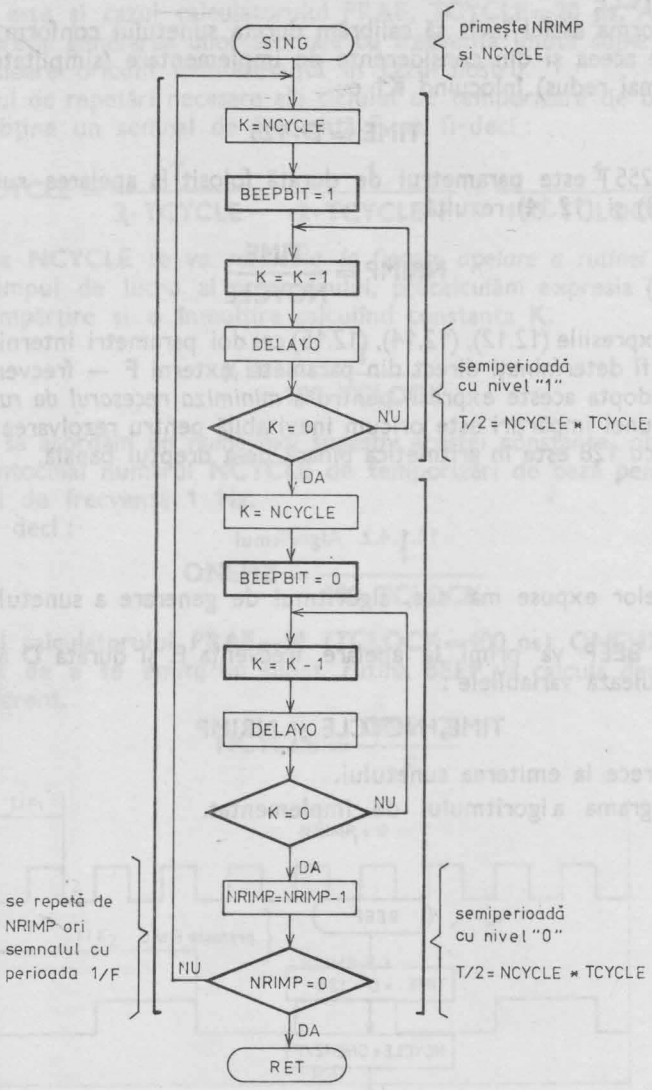


Fig. 12.19. Organigrama rutinei de emitere a sunetului : SING

12.1.4.3. Utilitare necesare

Din cele expuse în 12.1.4.1. și 12.1.4.2. rezultă necesitatea elaborării a 2 rutine de bază și anume :

- *rutină de temporizare* care să dureze o semiperioadă : **T/2 = NCYCLE · TCYCLE,** unde **TCYCLE** să dureze 50 de tacți procesor. O vom numi : **DELAY0.**
- *rutina de împărțire* a două numere binare exprimate pe 2 octeți, cu posibilitatea de rotunjire. O vom numi : **DIVIDE.**

Rutina DELAY0

■ Rutina va asigura *decrementarea unei variabile* **NCYCLE** de la o valoare inițială la **0**, astfel încît fiecare iterație să dureze 50 de tacți procesor. Gama de valori a variabilei **NCYCLE** este cuprinsă în plaja

NCYCLE = 1,25 pentru F = 20.000 Hz

NCYCLE = 1250 pentru F = 20 Hz

Pentru reprezentarea acestei variabile este necesară utilizarea unui registru dublu avînd 16 biți.

Se oferă următoarea secvență :

```
DELAY0 :                    ; DE = NCYCLE
         DEC   DE           ; 6
         JR    DELAY1       ; 12 — salt de temporizare
DELAY1 : JR    DELAY2       ; 12 — idem
DELAY2 : LD    A,E          ; 4 — test
         OR    D            ; 4 — pentru
         JR    NZ,DELAY0    ; 12 — DE = 0?
                            ; Total 50 tacți procesor
         RET                ; 10
```

În comentarii am marcat numărul de tacți ai fiecărei instrucțiuni folosite.

Rutina DIVIDE

Pentru împărțirea a două numere binare se pot imagina mulți algoritmi. Cazul cel mai simplu ar fi cel de a efectua *scăderi succesive* a **împărțitorului** din numărul **de împărțit,** pînă la obținerea unui număr negativ. Contorizînd scăderile se obține **cîtul** împărțirii. Această *metodă* are dezavantajul de a fi *banală*, și de a avea o durată de execuție variabilă în limite foarte largi, în funcție de numărul scăderilor de efectuat.

Pentru a putea aborda o metodă mai elaborată, *vom reconsidera* în cele ce urmează, *împărțirea a două numere zecimale*, semnalînd diferențele între aritmetica binară și cea zecimală.

Fie două numere zecimale cu maximum 4 cifre semnificative : **9734** și **0057**

Efectuăm împărțirea :

```
9734 : 57 = 170 — cîtul
57
---
403
399
---
0044 — rest
```

Procedura de sus, cunoscută tuturor, este prezentată doar pentru a fi criticată : ea beneficiază din plin de inteligența umană.

Primul și poate cel mai important pas îl reprezintă *calibrarea* celor 2 numere : oricine poate vedea în acest exemplu, că prima secvență de numere care poate

fi împărțită cu **57** este **97.** Se observă de asemenea la prima vedere, că cifra care reprezintă cîtul operației a **403 : 57** este **7.**

Această *ușurință* o vom avea ori de cîte ori vom dori să împărțim două *numere concrete.* Dacă dorim *în schimb* să împărțim două *numere oarecare*

$$a_3\ a_2\ a_1\ a_0 : b_3\ b_2\ b_1\ b_0$$

unde a_{0-3} și b_{0-3} pot fi orice cifră zecimală cuprinsă între **0** și **9,** va trebui să *elaborăm un algoritm* mai mecanic.

Să considerăm următorii regiștri :

REST — pentru generarea **restului** împărțirii

DEIMP — conține inițial **deîmpărțitul** la sfîrșitul operației el va conține **cîtul**

IMP — conține **împărțitorul**

Toți acești regiștri au aceeași lungime (număr de cifre semnificative) în cazul nostru 4. Pentru ușurință completăm și zerourile nesemnificative. Registrul **TEMP** este un registru cu o singură cifră semnificativă care va conține **cîtul** unei **împărțiri intermediare.** Valoarea acestuia va fi totdeauna cuprinsă între **0** și **9.**

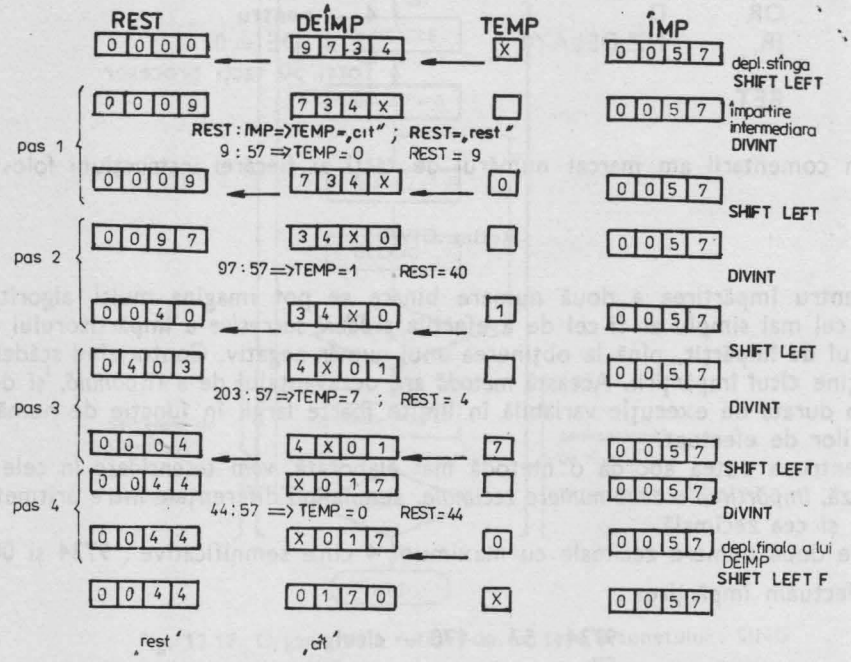


Fig. 12.20. Ilustrarea pașilor algoritmizați ai unei împărțiri zecimale (numere cu 4 cifre semnificative)

Am obținut astfel pe cale intuitivă o metodă ușor algoritmizabilă pentru împărțirea a două numere. Singura operație care ar putea ridica probleme la implementare este procedeul de efectuare a unei **împărțiri intermediare.** Ținînd însă cont de faptul că raportul celor două numere implicate în această împărțire intermediară este *întotdeauna mai mic decît un ordin de mărime,* rezultă că în arit-

metica binară problema poate fi rezolvată printr-o simplă *scădere* (Singurele cifre „**cît**" posibile fiind **0** și **1** nu există posibilitatea apariției unei cereri de iterație.)

Sintetizăm operațiile efectuate :

SHIFT LEFT : această operație *deplasează* cu o poziție *la stînga* conținutul regiștrilor **DEIMP** și **REST**, astfel ca cifra cea mai semnificativă din **DEIMP** să treacă pe poziția cea mai puțin semnificativă a lui **REST**.
În poziția cea mai puțin semnificativă a lui **DEIMP** este avansată cifra conținută în registrul **TEMP**.

DIVINT : este o *împărțire intermediară* între două numere a căror raport este totdeauna mai mic decît un ordin de mărime.
Se împarte **REST** cu **IMP. Cîtul** operației (o cifră) se depune în **TEMP**, iar restul în registrul **REST**.

SHIFT LEFTF : este o *deplasare la stînga finală*. Afectează doar **DEIMP** și **TEMP**. Scopul este de a genera "**cît**-ul" final al împărțirii. Conținutul lui **DEIMP** este rotit prin **TEMP** la stînga. Astfel ultimul **cît** intermediar se transferă din **TEMP** pe poziția cea mai puțin semnificativă a lui **DEIMP**, care devine astfel registru rezultat.

Putem construi organigrama din Fig. 12.21.

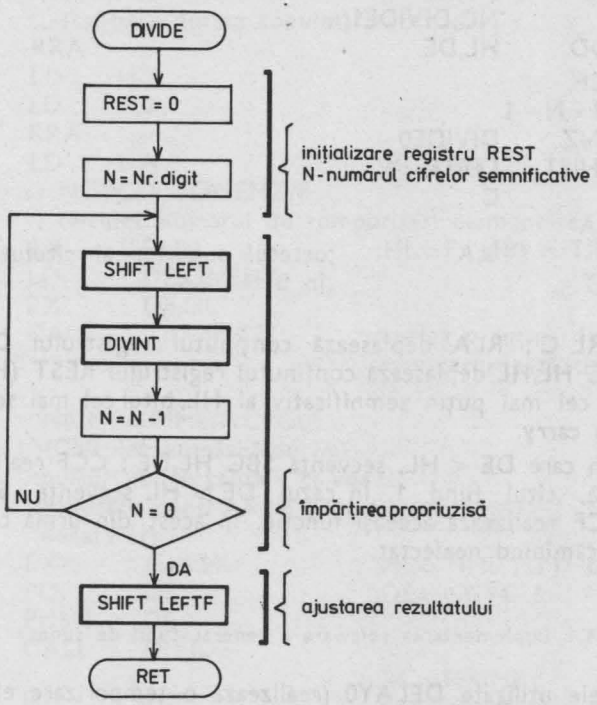


Fig. 12.21. Organigrama rutinei de împărțire : DIVIDE

Transpunerea acestui algoritm în limbaj de asamblare, pentru două numere a cîte 16 biți oferă, așa cum se va vedea, mult spațiu creativității și ingeniozității.

- Fie **A** și **C** **deîmpărțitul** (A octetul mai semnificativ).
- Fie **D** și **E** **împărțitorul** (D octetul mai semnificativ).
- Vom realiza o rutină care generează cîtul împărțirii **AC/DE** în registrul **BC**, restul obținîndu-se în **HL**.
- Rolul registrului **TEMP** folosit în exemplul de mai sus este preluat de flagul **Cy (carry)**.

Iată rutina :

```
DIVIDE :
            LD      A,H         ;transfer deîmpărțitul în
            LD      C,L         ;regiștri de lucru A și C
            LD      HL,0        ;inițializez restul
            LD      B,10H
            ;SHIFT  LEFT
DIVIDE0 :   RL      C
            RLA
            ADC     HL,HL
            ;DIVINT
            SBC     HL,DE
            JR      NC,DIVIDE1
            ADD     HL,DE
DIVIDE1 :   CCF
            ;N=N-1
            DJNZ    DIVIDE0
            ;SHIFT  LEFTF
            RL      C
            RLA
            LD      B,A         ;octetul superior al cîtului
            RET                 ;în B
```

- Secvența **RL C ; RLA** deplasează conținutul registrului **DEIMP (AÇ)** la stînga, iar **ADC HL,HL** deplasează conținutul registrului **REST (HL)** la stînga, înserînd pe bitul cel mai puțin semnificativ al **HL**, bitul cel mai semnificativ al **AC**, aflat acum în **carry**.
- În cazul în care **DE < HL**, secvența **SBC HL,DE ; CCF** realizează împărțirea intermediară, cîtul fiind 1. În cazul **DE > HL** secvența **SBC HL, DE ; ADD HL,DE ; CCF** realizează aceeași funcție. În acest din urmă caz **cîtul** este 0, **restul** din **HL** rămînînd neafectat.

12.1.4.4. Implementarea software a generatorului de sunet

Avînd modulele utilizate **DELAY0** (realizează o temporizare elementară de 50 tacți procesor) și **DIVIDE** (realizează împărțirea a două numere binare de 16 biți) constituite, putem trece la elaborarea rutinei **BEEP**.

■ Vom prelua cu mici modificări, rutina elabcrată de către *mat. Kiss Alexandru*, şi implementată pe calculatorul **PRAE—M** ca funcţie **BEEP** în **PRAE—BASIC V3.5.**

Folosind organigrama din fig. 11.18, programul se scrie uşor :

BEEP :

```
; subrutină generatoare de sunet
; după PRAE BASIC V3.5
;
; Parametri de intrare :
;    HL — conţine frecvenţa F în Hz
;     A — conţine un număr proporţional cu durata D
;
; regiştrii BC,DE, IX, IY — rămîn neafectaţi
;
                          ; salvez regiştri
PUSH BC
PUSH DE
PUSH HL
                            ;şterg Cy
SCF
CCF
 ; TIME=D*128
 ; (calibrez durata tonului)
RRA
LD    H,A
LD    A,0
RRA
LD    L,A                   ;HL=A*128
 ; NCYCLE=ONEHZ/F
; calculez numărul de temporizări elementare
EX    (SP),HL               ;HL=F, (SP) = TIME
LD      DE,ONEHZ
EX      DE,HL
CALL    COMPUTE             ;apelez o rutină de împărţire
                            ;care returnează cîtul rotunjit
                            ;în HL
;NRIMP=TIME/NCYCLE
;(Calculez numărul de im-
;pulsuri cu frecvenţa F, ce vor fi
;generate pentru a se obţine
;durata D)
EX      (SP),HL             ;HL=TIME,(SP)=NCYCLE
POP     DE                  ;DE=NCYCLE
PUSH    DE
CALL    DIVIDE
                            ;BC=NRIMP
INC     BC                  ;evit BC=0
;emit sunetul
POP     DE
```

```
        CALL    SING            ;emite NRIMP (BC) impulsuri
                                ;cu durata dată de NCYCLE(DE)
        POP     DE              ;restaurez regiştri
        POP     BC
        RET
```

■ Folosind rutina **DELAY0** elaborată la § 12.1.4.3. se poate scrie rutina **SING** a cărei organigramă se află în fig. 12.19.

```
SING :
        ;subrutină de emitere a sunetului
        ;Parametrii de intrare :
        ;       BC = NRIMP
        ;       DE = NCYCLE
        ;
        PUSH    DE              ;salvez NCYCLE
SING1 : LD      A,(WITNESS)     ;13 folosesc celula martor
        SET     BEEPBIT,A       ; 8 poziţionez 1
        POP     DE              ;10
        PUSH    DE              ;11
        OUT     (SYSOUT),A      ;11 emit 1
        CALL    DELAY0          ;17+DE*50+10
        LD      A,(WITNESS)     ;13
        RES     BEEPBIT,A       ; 8 poziţionez 0
        POP     DE              ; 10
        PUSH    DE              ; 11
        OUT     (SYSOUT),A      ; 11 emit 0
        CALL    DELAY0          ; 17+DE*50+10
        DEC     BC              ;  6
        LD      A,C             ;  4 NRIMP=NRIMP—1
        OR      B               ;  4
        JP      NZ,SING1        ; 10
        POP     DE              ; restaurez stiva
        RET
```

● Bucla program care emite **NRIMP** impulsuri, este cuprinsă între linia **SING1 :** şi instrucţiunea **JP NZ,SING1.** În dreapta fiecărei linii program am notat numărul de tacţi procesor aferenţi instrucţiunii respective. *Se observă că pe lîngă temporizările generate în rutina* **DELAY0** *(proporţionale* cu **NCYCLE),** *apar întîrzieri iminente datorită vehiculării datelor.* Frecvenţa sunetului emis va fi astfel mai mică decît cea teoretică. În § 12.1.4.5. vom prezenta abaterea frecvenţei reale de cea teoretică.

■ Ultima problemă de rezolvat a rămas elaborarea rutinei **COMPUTE,** care va efectua o împărţire binară de 16 biţi şi va rotunji cîtul.

Vom folosi rutina **DIVIDE** elaborată în § 12.1.4.3.

```
COMPUTE :
        ; rutină de împărţire cu rotunjirea cîtului
        ; parametri de intrare :
        ;       HL — deîmpărţitul
        ;       DE — împărţitorul
```

```
        ; parametri de ieşire :
        ;       HL — cîtul rotunjit
        CALL    DIVIDE
        PUSH    BC              ; salvez cîtul nerotunjit
        ADD     HL,HL           ; dublez restul
        CALL    COMP            ; dacă restul dublat
        POP     HL
        RET     C               ; este mai mic decît împărţitorul,
                                ; atunci cîtul nu se schimbă
        INC     HL              ; altfel, el este incrementat,
                                ; rotunjindu-se, rezultatul
        RET
COMP :
        ; rutina compară regiştri DE şi HL
        ; nu afectează nici un registru decît F
        OR      A               ; Cy=0
        PUSH    HL
        SBC     HL,DE
        POP     HL
        RET                     ; Cy=1 dacă DE > HL
```

■ Pentru ca modulul **BEEP** să fie complet, vom specifica valorile constantelor. Unde se poate le vom defini prin expresii, lăsînd evaluarea lor pe seama asamblorului. Conferim astfel un plus de elasticitate modulului realizat.

```
TCLOCK      EQU     400                 ; în nanosecunde
ONEHZ       EQU     50000/TCLOCK*200
EXTERNAL    BEEPBIT,SYSOUT,WITNESS      ; aceste constante se
                                        ; definesc în alt modul software,
                                        ; ierarhic superior.
```

12.1.4.5. Studiul abaterii frecvenţei reale de cea teoretică

Din implementarea enunţată rezultă faptul că frecvenţa sunetului emis, diferă de valoarea F_T stabilită la apelul rutinei **BEEP**. Ne propunem să determinăm valoarea acestei abateri.

Pentru a obţine perioada exactă T_R a sunetului emis în subrutina **SING**, vom însuma numărul de tacţi procesor aferenţi fiecărei instrucţiuni ai subrutinei :

$$N = 13+8+10+11+11+17+DE*50+10+13+8+10+11+11+17+DE*50+$$
$$+10+6+4+4+10$$

$$N = 182+100*DE$$

unde **DE** conţine numărul de iteraţii **NCYCLE**.

Rezultă perioada reală

$$T_R = (182+100*NCYCLE)*TCLOCK \qquad (12.16)$$

unde **TCLOCK** este perioada tactului procesor, 400 ns în cazul nostru.

Apare deci o abatere de la valoarea teoretică

$$T_T = 100 * NCYCLE * TCLOCK,$$

datorită întîrzierilor suplimentare care apar în timpul execuției rutinei **SING**. Totalul întîrzierilor se cifrează la :

$$\Delta T = 182 * TCLOCK \qquad (12.17)$$

ΔT reprezintă prima sursă de erori.

Cea de *a doua sursă de erori* apare la calculul numărului de iterații **NCYCLE**, în care *rezultatul* împărțirii **ONEHZ/F** este *rotunjit*.

Dorind să elaborăm un program **BASIC** pentru determinarea erorii relative a frecvenței sunetului emis notăm :

$$NCYCLE = INT\ (ONEHZ/F+0{,}5) \qquad (12.18)$$

unde **F** este frecvența teoretică impusă la apelul rutinei **BEEP**.

Frecvența reală va fi :

$$F_R = \frac{1}{\Delta T + 100 * INT(ONEHZ/F) * TCLOCK} \qquad (12.19)$$

Dorim să stabilim variația *erorii relative* a frecvenței :

$$\varepsilon = \frac{F_R - F_T}{F_T} * 100\% \qquad (12.20)$$

■ Următorul program **BASIC**, va stabili *eroarea relativă minimă* și *maximă*, și va desena *curba de variație a erorii relative* a frecvenței pe întregul domeniu audio (20 Hz — 20.000 Hz).

```
10  'Calculul erorii relative a funcției BEEP
20  TCLOCK= 400E-9
30  UNHZ=1/TCLOCK/100
40  DT=182*TCLOCK 'DELTA T
50  MIN=1 : F1=0 : MAX=0 : F2=0 'INITIALIZEZ MINIMUL ŞI
    MAXIMUL
60  DIM EPS (2000) 'ALOC SPAŢII TABELEI CU ERORI
70  I=0
80  CST = 100*TCLOCK 'PENTRU CREŞTEREA VITEZEI DE CALCUL
90  CLS : PRINT "LASAŢI-MA SA MA GINDESC"
100 FOR FT=20 TO 20000 STEP 10
110 EPS(I)= ((1/(DT+CST*INT(UNHZ/FT)))-FT)/FT
120 IF ABS(EPS(I)) < MIN THEN MIN=EPS(I) : F1=FT
130 IF ABS(EPS(I)) > MAX THEN MAX=EPS(I) : F2=FT
140 I=I+1
150 NEXT FT
160 CLS
170 PRINT USING "EROAREA MINIMA##.##% LA FT= ;"MIN*100 ; F1
180 PRINT
190 PRINT USING "EROAREA MAXIMA##.##% LA FT= ;MAX*100 ;
    F2
200 INPUT "DORIŢI GRAFICUL ERORII RELATIVE D/N" ; A$
210 IF A$="N" THEN END
220 GOSUB 270 'TRASAREA ŞI MARCAREA AXELOR
```

```
230  FOR I=2T02000
240  PLOT 80*LOG(I/2)/LOG(10), EPS(I)*400+10
250  NEXT I
260  GOTO 260
270  CLS 'TRASAREA şi MARCAREA AXELOR
280  'ABSCISA
290  PLOT 0,10 : DRAW255,10 :DRAW 252,8 :PLOT 255,10 : DRAW252,12
300  FOR I=1 TO4
310  PRINTAT (1+7*(I-1))*8,0 ;2*10^I ;
320  PLOT 80*(I-1),10 :DRAW 80*(I-1),8
330  NEXT I
340  PRINTAT 212,20 ;"FT" ;
350  'ORDONATA
360  FOR I=1 TO 5
370  PRINTAT 0,40*I+4 ; 10*I ;
380  PLOT 2,40*I+10 : DRAW 0,40*I+10
390  NEXT I
400  PLOT 0,10 : DRAW0,240 :DRAW2,237
410  PRINTAT 16,248 ; "-EPS%" ;
420  RETURN
```

Executînd acest program obţinem abaterile minime şi maxime ale funcţiei.

$\varepsilon_{min} = -0.14\ \%$ la frecvenţa $f_1 = 20$ Hz

$\varepsilon_{max} = -55.67\ \%$ la frecvenţa $f_2 = 20000$ Hz

Reprezentarea abaterii relative de frecvenţă o facem scalînd logaritmic axa frecvenţelor (vezi fig. 12.22).

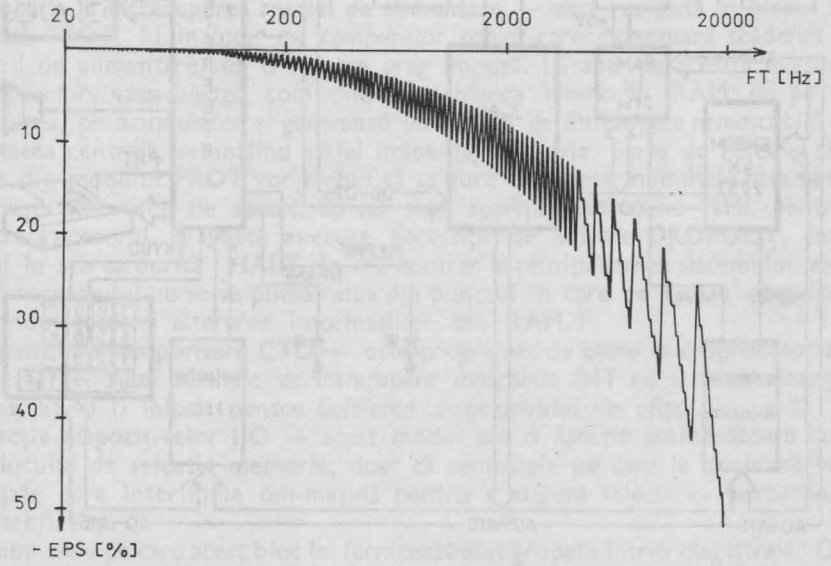


Fig. 12.22. Abaterea relativă a frecvenţei generate de BEEP

Putem conchide afirmînd că generatorul de sunet pe care l-am elaborat este destul de bun în gama frecvenţelor joase (20,2000 Hz), gamă în care abaterea relativă este mai mică de 10 %.

12.2. Structura hardware a echipamentului

Sintetizînd cele expuse în paragrafele precedente, *vom putea defini structura hardware* a echipamentului. Structura o vom organiza pe blocuri funcţionale, detailîndu-le pe alocuri la nivel de semnale electrice. Acesta este *momentul în care* cele două activităţi, *proiectare hardware şi software se despart*, urmînd ca ele să se desfăşoare cuasiindependent, pînă la momentul cheie, acela al punerii la punct a echipamentului.

Uşurinţa cu care se va face joncţiunea, depinde de *buna definire a detaliilor comune*. În cazul nostru aceste detalii se referă la cantitatea şi adresele blocurilor de memorie, precum şi la structura circuitelor de intrare/ieşire.

În fig. 12.23 am elaborat schema bloc a casei de marcat.

Analizînd fig. 12.23 putem identifica blocurile funcţionale majore ale echipamentului.

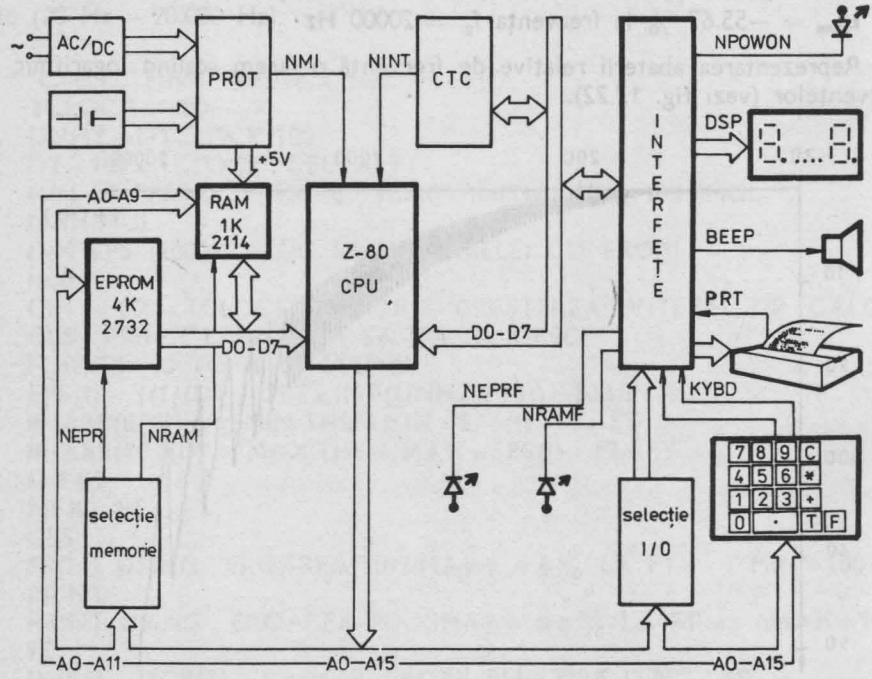


Fig. 12.23. Schema bloc a casei de marcat

1. **Unitatea centrală** — cuprinde microprocesorul **Z80**, amplificatoarele de semnal precum și oscilatorul pilot al sistemului.
2. **Memoria EPROM** — estimată grosier la **4 kbyte**, ea va fi constituită dintr-o singură capsulă de 4 kbyte : **I2732**. În ea vor rezida toate *programele casei de marcat*.
3. **Memoria RAM** — estimată la **1 kbyte** va fi realizată cu 2 circuite de memorie NMOS, de tipul **I2114**. În ea vor rezida *variabilele de lucru, stiva, zonele tampon ale software*-ului și *datele memorate de casa de marcat*, conform specificațiilor funcționale.

Datorită acestor din urmă date, *memoria* **RAM** va trebui să fie *nevolatilă*, prevăzîndu-se alimentarea ei de la baterie pentru cazul în care apar întreruperi ale rețelei de curent alternativ.

4. **Selecția memoriei** — format din circuite logice combinaționale, acest bloc funcțional are menirea să decodifice starea magistralei de adrese **A0—A15** și să genereze cele două semnale de selecție :
   - **NEPR** — va selecta memoria **EPROM** în spațiul de adrese ($0,0FFF_H$).
   - **NRAM** — va selecta memoria **RAM** în spațiul de adrese (8000_H, $83FF_H$).

   S-a păstrat intenționat o zonă „moartă" între sfîrșitul memoriei **EPROM** și începutul memoriei **RAM**, prevăzîndu-se astfel posibilitatea extinderii ușoare a hardwareului prin introducerea unei memorii **EPROM** mai mari (ex. **I2764**= **8k** sau **I27128**=16k).

Observație

În listingul din capitolul 17 precum și pe caseta **VISIBLE—Z80** programul pe care îl elaborăm împreună este generat pentru spațiul de adrese **7000H, 7FFFH** (partea dedicată pentru **EPROM**), iar zona **RAM** este inițializată la adresa **6C00H**. Explicația este simplă : la adresa 0, în ambele calculatoare (**PRAE** și **aMIC**) rezidă EPROM-urile proprii care conțin rutinele sistem și interpretorul **BASIC** al calculatoarelor. La adresa **8000H** se află însăși codul programului **VISZ80**. Menționăm că aceste abateri nu împietează cu nimic asupra înțelegerii prezentului studiu de caz.

5. **Protecția la întreruperea rețelei de alimentare** — este realizată în blocul funcțional **PROT**. El include un *comparator rapid*, care detectează scăderea tensiunii de alimentare sub o valoare prag impusă. La apariția acestui eveniment *comparatorul basculează*, comutînd alimentarea memoriei **RAM** de pe sursa de rețea, pe acumulator și *generează un semnal de întrerupere nemascabilă* către unitatea centrală, semnalînd astfel iminența de avarie. Sursa de rețea și circuitele din modulul **PROT** vor trebui să asigure tensiune, în limitele parametrilor impuși, încă timp de aprox. 50 **ms** după apariția semnalului **NMI**, pentru ca microprocesorul să poată executa secvența de salvare **DROPOUT**, intrînd apoi în starea oprită : **HALT**. În caz contrar la reinițializarea sistemului, execuția programului nu se va putea relua din punctul în care ea fusese abandonată, riscîndu-se astfel alterarea informațiilor din **RAM**.
6. **Circuitul de temporizare CTC** — este programat de către microprocesor astfel încît să genereze *semnale de întrerupere mascabile* **INT** cu o intermitență de 2 **ms**. El va fi folosit pentru *baleierea dispozitivului de afișaj*.
7. **Selecția dispozitivelor I/O** — acest modul are o funcție asemănătoare cu cea a blocului de selecție memorie, doar că semnalele pe care le generează vor fi dirijate spre interfețele om-mașină pentru a asigura selectarea porturilor de intrare/ieșire.

Semnalele pe care acest bloc le furnizează sînt grupate într-o magistrală. Detalierea lor se poate vedea în fig. 11.24.

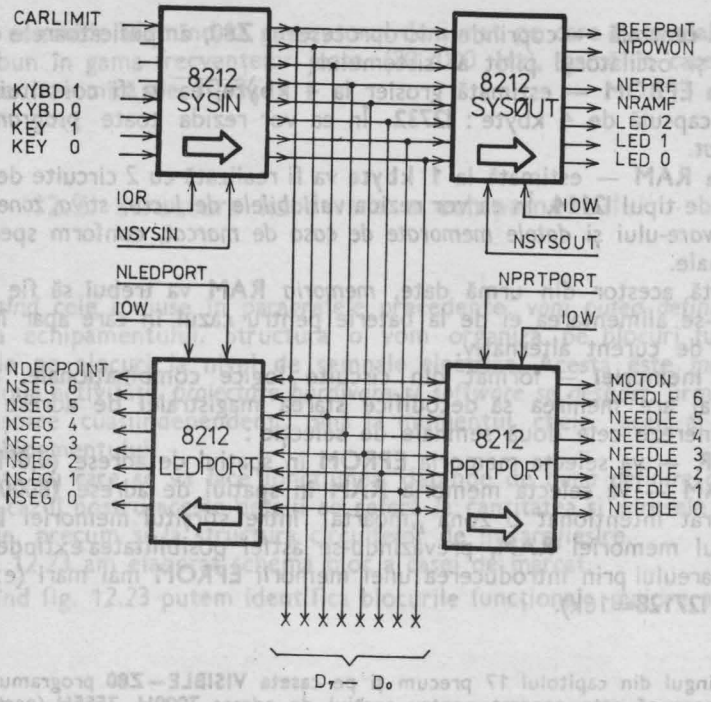


Fig. 12.24. Structura şi semnalele circuitelor de intrare/ieşire

- **IOR** — activ în starea "1" ; semnalează execuţia unui ciclu maşină **IN.**
- **IOW** – activ în starea "1" ; semnalează execuţia unui ciclu maşină **OUT.**
- **NSYSIN** — este o *valoare decodificată* a liniilor **A0—A7** şi asigură selectarea portului de intrare **SYSIN.**
- **NSYSOUT** — este o *valoare decodificată* a liniilor **A0—A7** şi asigură selectarea portului de ieşire **SYSOUT.** Poate fi identic cu **SYSIN.**
- **NLEDPORT** — este o *valoare decodificată* a liniilor **A0—A7** şi asigură selectarea portului de comandă a dispozitivului de afişaj **LEDPORT.**
- **NPRTPORT** — este o *valoare decodificată* a liniilor **A0—A7** şi asigură selectarea portului de comandă a imprimantei **PRTPORT.**

8. INTERFETE — este un *modul complex* care cuprinde pe lîngă porturile de intrare/ieşire şi circuite electronice formate din componente discrete pentru ajustarea nivelului energetic şi de tensiune a tuturor semnalelor care realizează schimbul de informaţii între **UC** şi periferice.

Din punctul de vedere al obiectivului pe care acestă carte şi-l propune, electronîca discretă de interfaţă este neinteresantă. Vom insista în schimb asupra structurii logice, asupra semnalelor porturilor de I/O.

În fig. 12.24 regăsim cei 4 porţi de intrare/ieşire precum şi semnalele de comandă şi stare pe care le-am prevăzut.

■ **SYSIN — este portul de intrare sistem.** Semnalele care pot fi citite prin acest port sînt :

● **KEY0** și **KEY1** sînt cele 2 chei a căror prezență a fost impusă în specificația constructivă (vezi pct. C.8 din § 11.1)

● **KEY0** este cheia operatorului

● **KEY1** — este cheia de control

Ambele semnale sînt active în starea **"1"**

● **KYBD0** și **KYBD1 (D2** și **D3)** — sînt cele două linii ale tastaturii. Apăsarea unei taste se materializează prin apariția unui nivel **"0"** pe unul din cele două linii. Următorii *trei biți* ai portului au fost lăsați *neutilizați*, prevăzîndu-se astfel *posibilitatea extinderii* liniilor de tastatură, pentru o eventuală dezvoltare a tastaturii pînă la 40 de taste.

● **CARLIMIT (D7)** — este semnalul generat de *detectorul capăt de cursă* al imprimantei. Este activ în starea **"1"**.

■ **SYSOUT** — este **portul de ieșire sistem.** Conținutul lui va fi *memorat permanent* în memoria **RAM,** în celula martor **WITNESS.** Semnalele care pot fi generate prin acest port sînt :

● **LED0—LED2 (D0—D2)** — reprezintă contorul de selecție al unuia din cele 8 **LED**-uri de afișaj. Valorii binare *000* îi corespunde **LED**-ul din *extrema dreaptă*, iar valoarea *111* selectează **LED**-ul din *extrema stîngă* a dispozitivului de afișaj.

● **NRAMF (D3)** — este un semnal activ în starea **"0"**, el *va „aprinde"* o diodă luminiscentă dispusă pe circuitul imprimat al unității centrale, *semnalînd* astfel personalului de service *eroare la memoria* **RAM,** eroare detectată în secvența de inițializare a casei de marcat.

● **NEPRF (D4)** — este un semnal activ în starea **"0"**, funcția lui este asemănătoare cu cea a semnalului **NRAMF,** cu deosebirea că el *semnalează eroare la memoria* **EPROM.** Următorul bit **(D5)** s-a păstrat ca rezervă pentru o semnalizare suplimentară, ce se va putea introduce pe viitor.

● **NPOWON (D6)** — este un semnal activ în starea **"0"** ; el *va „aprinde"* o diodă luminiscentă amplasată pe panoul frontal al casei de marcat, *semnalînd prezența tensiunii* de alimentare. Ba chiar mai mult, operatorul va ști că dispozitivul *trăiește*, din moment ce pînă la „aprinderea" **LED**-ului de prezență tensiune, microprocesorul va fi executat o serie întreagă de rutine.

● **BEEPBIT(D7)** — este *linia* pe care se vor genera *semnale sonore*, formînd pe cale software trenuri de impulsuri de durată și frecvență dorită.

■ **LEDPORT** — este un **port de ieșire** care asigură comanda dispozitivului de afișaj. Semnalele lui de ieșire sînt cablate la toate elementele de afișaj, formînd astfel magistrala locală a dispozitivului de afișaj. Toate semnalele sînt active în starea **"0" NSEG0—NSEG6 (D0—D6)** — reprezintă biții de activare a segmentelor, iar **NDECPOINT** „aprinde" punctul zecimal.

■ **PRTPORT** — este un **port de ieșire,** avînd rolul de a comanda imprimanta. Toate semnalele lui sînt active în stare **"1"**.

NEEDLE0—NEEDLE6(D0—D6) activează **acele,** iar **MOTON** pornește și/sau oprește **motorul** de antrenare a capului de imprimare.

Odată cu aceste elemente, considerăm terminată definirea structurii hardware a casei electronice de marcat.

Măsura în care această structură va rezista „furtunilor" ce apar pe parcursul proiectării detailate a hardware-ului și mai ales în faza de punere la punct a echipamentului, *depinde* în bună măsură *de conștiinciozitatea* cu care ea a fost elaborată, precum *și de experiența proiectantului.*

13

DEFINIREA STRUCTURILOR DE DATE

Avînd hardware-ul definit, *vom reanaliza* de mai multe ori *specificația funcțională*, pentru a ne putea forma o *imagine de ansamblu* asupra întregului software pe care urmează să-l realizăm. Așa cum am arătat în Cap. 10, *definirea structurilor majore de date, identificarea principalelor variabile* ale unui program, trebuie să preceadă demararea elaborării programelor în sine.

Pe baza experienței acumulate se poate afirma că *durata de elaborare și implementare* a componentelor software *poate fi substanțial redusă*, dacă ea este precedată de un efort suplimentar, pentru definirea elementelor sus menționate.

În cele ce urmează vom încerca să *delimităm* toate *structurile care se pot întrezări* pe baza specificațiilor constructive și funcționale, precum *și cele impuse* de periferice, a căror tratare s-a prezentat în Cap. 12.

La sfîrșitul acestei activități, cea de definire a principalelor structuri de date, *vom fi în măsură să elaborăm o primă schiță a fluxului de date în casa de marcat*, schiță pe care o vom reconsidera la sfîrșitul lucrării, după ce vom fi elaborat întregul software.

13.1. Structuri de bază

Dacă pentru prezentarea celor mai eficienți algoritmi de înmulțire, împărțire, sau a celor de grafică bi-și tridimensională s-au scris zeci de cărți, și bibliografia aferentă structurilor de date este deosebit de bogată. *Numim* **date**, *absolut toate informațiile care deși nu reprezintă program (cod) executabil, sînt incluse într-un program, sau se generează pe parcursul execuției acestuia menționînd că existența lor este absolut necesară pentru programul în cauză. Toate informațiile referitoare la natura datelor, la codul și forma lor de reprezentare, precum și la modul în care ele sînt vehiculate, determină structura datelor.*

Noțiunea structurilor de date este o clasă deschisă, care se îmbogățește mereu, putîndu-se imagina o infinitate de reprezentări și manevre. Este greu să aplici clasificări pe mulțimi infinite. Rolul de spirit ordonator al acestei „jungle" îl pot avea, înainte de toate, tipurile de aplicații. Într-adevăr, în jurul lor (le-am putea numi focare) s-au cristalizat și se cristalizează mereu soluții tip. Aceasta este

calea de prezentare aleasă în majoritatea lucrărilor de strictă specialitate, dedicate structurilor de date. O vom urma și noi, prezentînd — în ordinea importanței lor — structurile de date legate de virtuala casă de marcat.

13.1.1. Codurile interne

Înainte de toate vom stabili codurile de reprezentare internă pentru toate caracterele tastaturii, a celor afișabile, precum și a celor care se vor imprima. Calea cea mai rapidă ar fi aceea de-a adapta un set standard. De exemplu, cel mai răspîndit set de coduri, cel **ASCII** (American Standard Code for Information Interchange).

Această soluție ar prezenta avantajul universalității și unei anumite portabilități. Ea ar fi deosebit de interesată în cazul în care casa de marcat ar trebui să fie înglobată într-un sistem informațional complex.

Analizînd specificația dispozitivului de afișaj, a tastaturii, precum și formatul codurilor, ajungem la concluzia că aplicația noastră nu solicită decît un set restrîns din totalitatea caracterelor incluse în standardul **ASCII**. De aceea *ne vom permite să elaborăm un sistem propriu de coduri.*

Sîntem conștienți că această decizie s-ar putea să fie contestată de unii, dar o menținem, dorind să-i *oferim cititorului un exemplu de acțiune* în acest sens.

Iată setul de caractere propus, și codurile aferente :

Cod	Car	Cod	Car	Cod	Car	Cod	Car	Cod	Car
00	0	08	8	10	A	18	Q	20	spațiu
01	1	09	9	11	C	19	P	21	:
02	2	0A	.	12	D	1A	R	22	—
03	3	0B	FUNC	13	E	1B	S		
04	4	0C	+	14	I	1C	T		
05	5	0D	*	15	L	1D	U		
06	6	0E	TOTAL	16	M	1E	V		
07	7	0F	CLEAR	17	N	1F	Z		

Tab. 13.1. Codurile interne ale casei de marcat

În primele două grupe verticale din Tab. 13.1 se regăsesc codurile caracterelor tastaturii. Faptul că, *codurile interne alocate cifrelor* **0—9**, *coincid cu valoarea binară cifrei reprezentate*, nu este o întîmplare. Astfel ne scutim de efortul realizării unei conversii suplimentare pe cale software. Intenționat am acordat *următorul cod* **($0A_H$)**, *semnului* "." **(punct)**, deoarece el va fi folosit în majoritatea cazurilor ca punct zecimal. Adiacența codului său cu codurile cifrelor va putea ușura pe alocuri implementarea rutinelor de conversie zecimal — binar, binar — zecimal. *Ordinea alocării codurilor pentru* celelalte cinci caractere ale tastaturii

(FUNC, +, *, TOTAL, CLEAR) *este neinteresantă.* Le-am preluat bineînțeles pe acele care rezultă din structura logică a tastaturii, așa cum le furnizează rutina **INKEY.** In continuare am alocat coduri literelor utilizate, așezate în ordine alfabetică (valorile sînt hexazecimale).

În ultima grupă am inclus caracterul „spațiu" și cele două semne de punctuație " :" (două puncte) respectiv "—" (minus). Faptul că, *codul alocat* caracterului **"spațiu" (20_H),** este *identic* cu codul aceluiași caracter în setul **ASCII,** este o *pură întîmplare.*

13.1.2. Generatorul de caractere (s gmente) pentru dispozitivul de afișaj : LEDGEN

Așa cum s-a arătat în paragraful 11.2.2., vizualizarea numerelor se va face pe un dispozitiv de afișaj care înglobează 8 elemente de afișaj **(LED**-uri cu 7 segmente). Fiecărei cifre afișabile **(0—9)** îi corespunde un cuvînt binar de 7 bit unic determinat. Aplicînd acest cuvînt la intrările **NSEG0—NSEG6** ale **LED**-ului, el va determina "aprinderea" segmentelor a căror biți de comandă au valoarea "0".

În fig. 12.5. am ilustrat modul de determinare a cuvintelor de comandă pentru cifrele **9** și **5,** aceasta din urmă avînd în exemplul prezentat, punctul zecimal activat. *Reținem că segmentul activ înseamnă bit de comandă* **"0"**

Procedînd în mod similar pentru toate cele zece cifre vom obține codurile de comandă ale segmentelor. Grupîndu-le într-o listă, realizăm o tabelă pe care o numim *generator de caractere a dispozitivului de afișaj.* Îi atașăm numele simbolic **LE DGEN.** Această tabelă de **10 octeți** o vom înscrie în **EPROM,** începînd de la adresa **LEDGEN.** Codurile de comandă a segmentelor fiind așezate în ordinea crescătoare a cifrelor, *referirea* oricăruia dintre ele *se va* putea *face* simplu, *adăugînd la adresa de bază* **LEDGEN** însăși *codul intern al cifrei* dorite. *Octetul citit* din memorie, de la adresa astfel obținută, *va fi codul de comandă segmente al cifrei* respective. Acest octet poate fi depus pe magistrala locală a dispozitivului de afișaj pentru a vizualiza pe un **LED** cifra în cauză.

În fig. 13.1 redăm structura și conținutul generatorului de caractere **LED-GEN.**

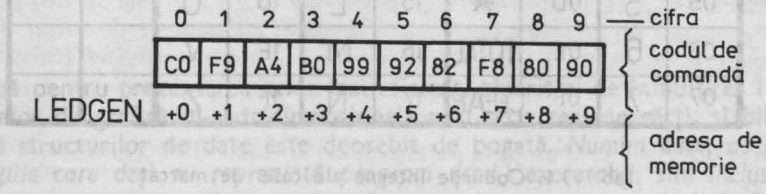


Fig. 13.1. Generatorul de caractere pentru dispozitivul de afișaj : LEDGEN

Codurile de comandă din **LEDGEN** sînt exprimate în sistem hexazecimal. Se poate observa că punctul zecimal nu este activat, în niciunul din cele **10** *coduri de comandă.* „Aprinderea" punctului zecimal va cădea în sarcina software-ului.

13.1.3. Generatorul de caractere (puncte) pentru imprimantă : PRTGEN

Din descrierea interfeței de imprimantă (paragraf 11.2.3) reținem că *rutina de imprimare a unui caracter*, **PRTCHAR**, folosește codul intern al caracterului, cu ajutorul căruia *localizează* în memorie *cinci octeți de comandă ace*, imprimînd succesiv cele cinci coloane ale caracterului. Astfel din mișcarea corelată a capului de imprimare și a acelor, se poate obține — într-o matrice de **5 × 7** puncte desenul oricărui caracter imprimabil.

În fig. 13.2. exemplificăm modul în care utilizatorul își poate defini forma caracterelor; stabilind în același timp și codurile de comandă ale acelor. *Reamintim* faptul că *un bit* de comandă cu valoarea **"1"**, înseamnă *punct imprimat*. Tot aici specificăm că *imprimarea* se face *de la dreapta la stînga*.

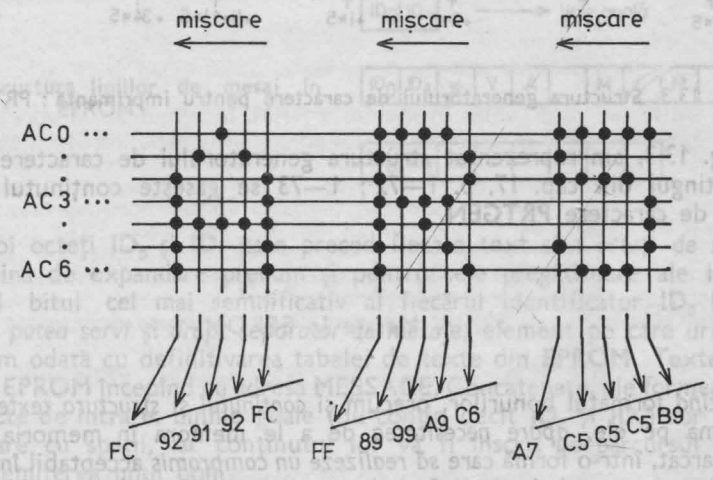


Fig. 13.2. Exemplu pentru constituirea generatorului de caractere a imprimantei : PRTGEN

Grupînd octeții de comandă ai acelor (aferente caracterelor) în ordinea crescătoare a codurilor interne, *obținem generatorul de caractere* **PRTGEN.** Amintim faptul că pe *bitul* D_7 al portului de ieșire **PRTPORT** se *comandă mișcarea motorului* de antrenare a capului de imprimare. *Acest bit va avea valoarea* **"1"** în toți octeții de comandă **PRTGEN,** deoarece acționarea acelor cu motorul oprit nu are sens.

Pentru a ușura localizarea codurilor de comandă ace, aferente unui caracter, în **PRTGEN** *vom rezerva cîte cinci octeți și celor trei coduri neimprimabile* : **FUNC** [$0B_H$], **TOTAL** [$0E_H$] și **CLEAR** [$0F_H$]. Conținutul acestor zone este lipsit de interes. Astfel obținem o listă care o vom stoca în **EPROM,** începînd cu adresa **PRTGEN.** Lista va avea **35 de puncte de intrare,** ale căror adrese se obțin ușor, aplicînd formula :

$$\text{ENTRY} := \text{PRTGEN} + I * 5 \qquad (13.1)$$

Fiecare intrare reprezintă adresa de început a grupei de comandă ace, aferentă unui caracter. În cadrul grupei, octeții sînt depuși în ordine inversă, începînd cu coloana din extrema dreaptă a desenului caracterului, datorită faptului că imprimarea are loc *de la dreapta la stînga.*

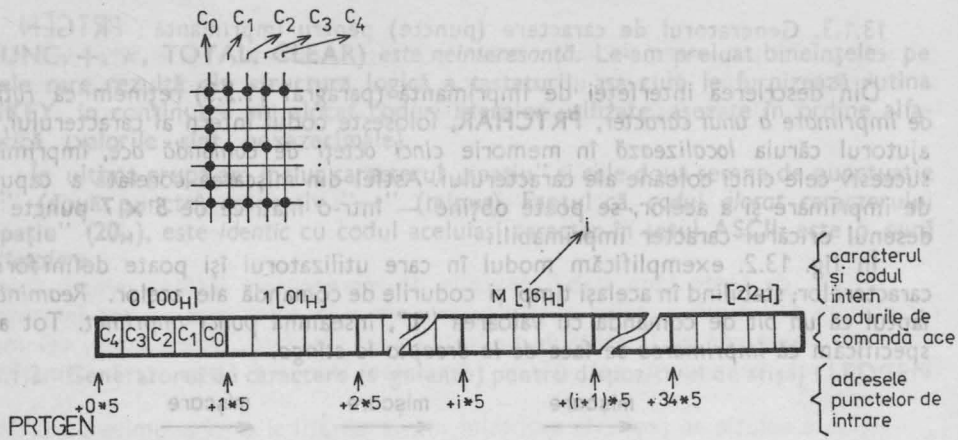


Fig. 13.3. Structura generatorului de caractere pentru imprimantă : PRTGEN

În fig. 13.3. am reprezentat structura generatorului de caractere **PRTGEN.** În listingul din cap. 17, p. 1—72 ; 1—73 se găsește conținutul întregului generator de caractere **PRTGEN.**

13.1.4. Mesaje în EPROM

Analizînd formatul bonurilor, precum și *conținutul* și *structura textelor* care se vor imprima pe ele, *apare necesitatea* de a le memora în memoria **EPROM** a casei de marcat, într-o formă care *să realizeze un compromis* acceptabil *între necesar de memorie* pentru tabelare *și efort de programare.* Cele două deziderate sînt contradictorii și nu pot fi minimizate concomitent.

Soluția banală, care ar necesita un efort de implementare minim ar fi aceea de a stoca în **EPROM** mesajele fiecărui bon în parte, structurate conform cerințelor de imprimare impuse prin formatul bonurilor. Soluția aceasta se respinge datorită faptului că *ar consuma* o cantitate substanțială de memorie EPROM :
4 tipuri * 9 linii * 15 caractere = 675 octeți, *aproape* **16%** *din totalul de spațiu* **EPROM.**

Cealaltă extremă o reprezintă soluția de a memora pur și simplu fiecare mesaj în parte **(TOTAL, CASA NR, UNIT NR., * VA MULȚUMIM *,** etc.) reducînd astfel *necesarul de octeți* pentru mesaje la **75.** În acest caz *se poate însă întrezări o încîlcire peste măsură a programelor* de editare a bonurilor, programe care vor trebui să rezolve generarea formatului de bon impus.

În această situație soluția de compromis devine necesară. Propunem următoarea procedură : în EPROM i se va rezerva fiecărui tip de rînd un cîmp de informații. Acest cîmp va fi completat cu informații ajutătoare pentru programele de emitere a bonurilor, precum și cu textul mesajului (doar în rîndurile în care ele vor trebui să existe). Informațiile ajutătoare se vor referi la tipul de bon. *Doi octeți ar fi suficienți* în acest sens. Caracterele spațiu cuprinse între ultimul caracter de text și sfîrșitul liniei nu se înscriu în **EPROM.**

Se va elabora o rutină **EXPAND** care *va transfera la trezire mesajele din* **EPROM** în **RAM**, completîndu-le cu spațiile necesare. Zona în care aceste mesaje se transferă o numim zonă de editare pentru imprimare. Asupra structurii ei vom reveni în acest capitol. Specificăm acum doar atît că, în această zonă, fiecare linie va avea deja structura „oglindă" a imaginii ei de pe bon.

În **EPROM**, *liniile vor avea lungimi diferite*, în funcție de existența sau inexistența unei secvențe de text pe rîndul respectiv. Este evident că dacă lungimea a două texte diferă, atunci și lungimea liniilor respective în **EPROM** va diferi.

In zona de editare pentru imprimare din memoria **RAM** *toate rîndurile vor avea aceeași lungime*, de **17** caractere.

Structura liniilor de mesaj în **EPROM** o redăm în fig. 13.4.

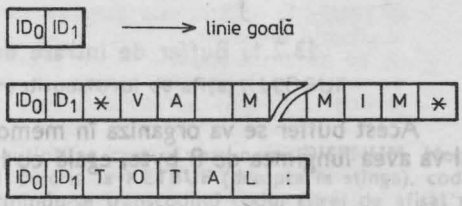


Fig. 13.4. Structura liniilor de mesaj în EPROM.

Cei doi octeți ID_0 și ID_1 care preced fiecare text sînt *octeți de identificare* pentru rutina de expandare precum și pentru cele pregătitoare ale imprimării. Poziționînd bitul cel mai semnificativ al fiecărui identificator ID_0 la valoarea **"1"**, el *va putea servi* și *drept separator de mesaje*, element pe care urma oricum să-l stabilim odată cu definitivarea tabelei de texte din **EPROM**. Textele le vom amplasa în **EPROM** începînd cu adresa **MESSAGE**. Concatenate, ele formează o listă cu **14 puncte de intrare**. Liniile goale (nu conțin decît ID_0 și ID_1) vor fi umplute la expandare cu spații, iar conținutul lor va fi înscris pe parcursul operațiilor legate de emiterea unui bon.

Adoptînd această structură de informație *lungimea tabelei de texte* în **EPROM** *va fi de* **114 octeți**, iar cea a zonei de editare în **RAM** $14 \times 17 = 238$ **octeți**.

Apreciind faptul că rutina de expandare precum și cea de pregătire a imprimării nu vor depăși lungimea de **100 octeți** considerăm obiectivul propus, cel de a găsi un compromis între necesar de memorie și efort de programare, ca fiind atins.

În listingul din Cap. 17, pag. 1—70, 1—71, se găsește conținutul exact al tabelei de texte din **EPROM (MESSAGE)**. În dreptul fiecărei linii conținutul ei se indică în clar cu majuscule, sub forma unui comentariu. În dreptul liniilor vide se menționează (în paranteze, cu litere mici) destinația lor, adică informația care se va înscrie în ele pe parcursul operațiilor legate de emiterea unui bon.

13.2. Zone de manevră

Structurarea programelor este adesea înlesnită prin folosirea unor *zone tampon de memorie, pentru vehicularea datelor*. Aceste zone de manevră poartă numele de *buffer*. Constituirea lor permite o delimitare mai clară a activităților diverselor

module de software. Astfel *se poate urmări mai uşor fluxul informaţional* realizat în orice echipament. Utilizarea bufferelor este caracteristică atît rutinelor de intrare/ieşire cît şi modulelor software ierarhic superioare.

Pornind de la aceste premise ne propunem ca, codul fiecărei taste citite să fie depus într-un buffer **(KEYBUF)**. Aşa vom proceda, şi cu caracterele care urmează-să fie vizualizate (se depun în **LEDBUF)** precum şi cu cele de imprimat, **(PRTBUF)**. Aceste manevre vor fi efectuate de către modulele software imediat superioare rutinelor de tratare fizică a interfeţelor, prezentate în Cap. 12.

În afara acestor buffere constituite pentru deservirea interfeţelor vom defini şi zone de manevră, care vor servi comunicaţia cu module software ierarhic superioare.

13.2.1. Buffer de intrare date de la tastatură : KEYBUF

Acest buffer se va organiza în memoria **RAM**, începînd de la adresa **KEYBUF**, şi va avea lungimea de **8 byte**, egală cu lungimea dispozitivului de afişaj.

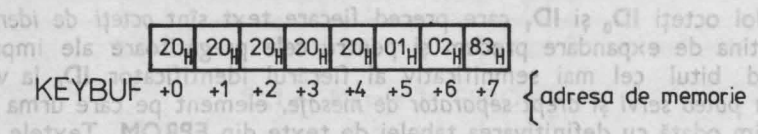


Fig. 13.5. Buffer de intrare date de la tastatură : KEYBUF

Trăsături :

- **KEYBUF** conţine *codurile interne* ale ultimelor 8 cifre tastate ;
- *punctul zecimal* "." se reprezintă **setînd** (valoarea "1") bitul cel mai semnificativ al ultimei cifre tastate **(KEYBUF+7)** ;
- poziţia **KEYBUF+7** este *cea mai puţin semnificativă* şi apare pe dispozitivul de afişaj în extrema dreaptă ;
- *încărcarea* **KEYBUF** se va face cu o rutină pe care o vom numi **STORENUM**. Ea va deplasa conţinutul întregului buffer cu o poziţie la stînga, noul caracter (cod intern) fiind introdus pe poziţia **KEYBUF+7** ;
- cu ocazia deplasării caracterul cel mai semnificativ (de pe poziţia **KEYBUF+0**) se va pierde.

13.2.2. Buffer de ieşire pentru dispozitivul de afişaj : LEDBUF

LEDBUF se va organiza în memoria **RAM**, începînd de la adresa **LEDBUF**, şi va avea lungimea de **8 byte**, egală cu lungimea dispozitivului de afişaj.

Trăsături :

- **LEDBUF** conţine *codurile* binare *de comandă segmente*, pentru cele 8 LED-uri cu 7 segmente şi punct zecimal ;
- *segmentul aprins* apare ca bit de valoare "0", iar cel stins cu valoarea "1" ;
- *punctul zecimal* se reprezintă înscriind valoarea zero în bitul cel mai semnificativ (D_7) al octetului corespunzător **LED**-ului dorit ;

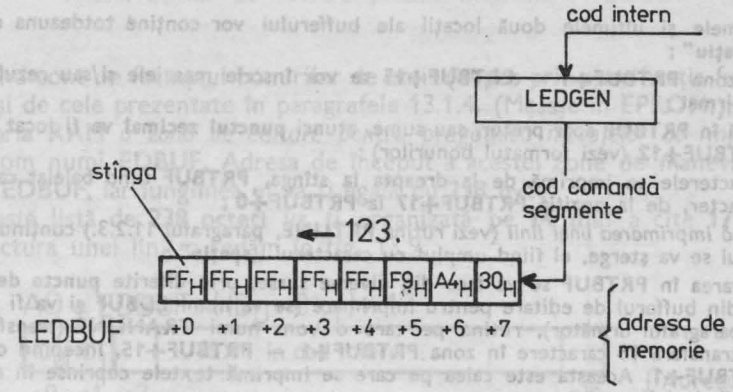


Fig. 13.6. Buffer de ieşire pentru dispozitivul de afişaj : LEDBUF

- *Încărcarea* **LEDBUF** se va face cu o rutină pe care o vom numi **DISPNUM**. Mecanismul de încărcare seamănă cu cel descris la **KEYBUF** (dreapta la stînga), codul introdus pe poziţia **LEDBUF+7** obţinîndu-se transcodînd codul cifrei de afişat cu ajutorul generatorului de caractere (segmente) a dispozitivului de afişaj **LEDGEN** ;
- poziţia **LEDBUF+7** este cea mai puţin semnificativă şi apare pe dispozitivul de afişaj în poziţia extremă dreaptă ;
- *transferul* conţinutului **LEDBUF** la *dispozitivul de afişaj* va fi efectuat de rutina de deservire a întreruperilor mascabile **INT**, care va fi activată din 2 în 2 **ms** ;
- deosebim un *caz special*, în care încărcarea **LEDBUF** nu se va face conform procedurii enunţate mai sus.

În cazul folosirii oricărei proceduri, declanşată prin apăsarea succesivă a tasei **FUNC** (1—7 tastări) poziţia **LEDBUF+0** (extrema stîngă a dispozitivului de afişaj) va fi folosită ca numărător de tastări succesive, **FUNC** (vezi specificaţia funcţională **F2.0.**).

În acest caz *înscrierea codului de afişat se va face direct pe poziţia* **LEDBUF+0**, fără a deplasa conţinutul bufferului. Această operaţie va fi efectuată de o rutină pe care ne propunem să o numim **CNTFUNC**.

13.2.3. Buffer de ieşire pentru imprimantă : PRTBUF

PRTBUF va fi organizat în memoria RAM, începînd cu adresa **PRTBUF**, şi va avea lungimea de **18 byte**. În acest buffer se va genera imaginea (reprezentată în coduri interne) a unui rînd care se va imprima cu ajutorul rutinei deja prezentate **PRT18C**.

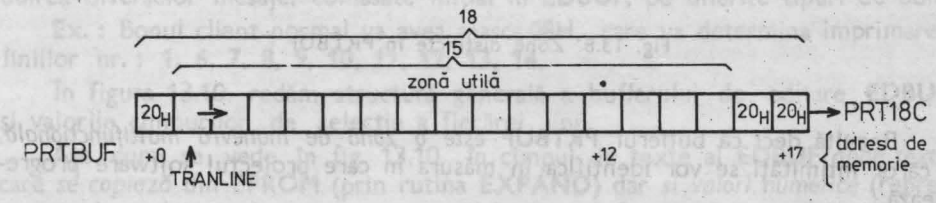


Fig. 13.7. Structura generală a bufferului de ieşire la imprimantă : PRTBUF

Trăsături :

- primele și ultimele două locații ale bufferului vor conține totdeauna caracterul **„spațiu"** ;
- în zona **PRTBUF+1 — PRTBUF+15** se vor înscrie mesajele și/sau rezultatele de impirmat ;
- dacă în **PRTBUF** apar prețuri sau sume, atunci **punctul zecimal** va fi locat la adresa **PRTBUF+12** (vezi formatul bonurilor) ;
- caracterele se imprimă de la dreapta la stînga, **PRTBUF** fiind baleiat caracter cu caracter, de la poziția **PRTBUF+17** la **PRTBUF+0** ;
- *după imprimarea unei linii* (vezi rutina **PRTLINE**, paragraful 11.2.3.) conținutul bufferului se va șterge, el fiind umplut cu caracterul **„spațiu"** ;
- intrarea în **PRTBUF** se va face din diverse surse, prin diferite **puncte de intrare** :
  a) din bufferul de editare pentru imprimare (se va numi **EDBUF** și va fi prezentat în paragraful următor), rutina pe care o vom numi **TRANLINE** (transferă linie) va transfera 15 caractere în zona **PRTBUF+1 — PRTBUF+15**, începînd cu poziția **PRTBUF+1**. Aceasta este calea pe care se imprimă textele cuprinse în antetul și la sfîrșitul bonurilor.
  b) pe parcursul operării se vor imprima însă și linii individuale cuprinzînd coduri de sortiment și prețuri, precum și un marcaj locat în prima coloană din dreapta, privind tipul operației efectuate (vezi formatul bonurilor și specificația funcțională **F.1.8.**). **Încărcarea bufferului** pentru efectuarea acestor operații se va face după tastarea **tastei "+" printr**-o procedură pe care ne propunem să o denumim **OPFORBUY (operații pentru** client). În această conjuncturâ **PRTBUF** (sau zone ale lui) va fi **încărcat direct** prin diverse rutine a căror structură exactă nu se poate întrezări **în acest moment** ;
- putem identifica cîteva *zone distincte* :
  — **PRTBUF+1 — PRTBUF+2** — va conține **codul de sortiment**
  — **PRTBUF+4 — PRTBUF+14** — va conține **prețuri** (sau **totaluri**)
  — **PRTBUF+12** — va conține **punctul zecimal**
  — **PRTBUF+15** — va conține **un semn (+, —, A sau ∗)** specific operației efectuate ;
- știind că după apăsarea tastei "+", dispozitivul de afișaj va trebui să conțină suma curentă a clientului, iar pe imprimantă se va imprima ultimul preț (sau valoare) introdus, precum și în ideea de a reduce numărul zonelor de manevră **RAM**, *ne propunem ca* **PRTBUF** (*zona* **PRTBUF+4 — PRTBUF+14**) *să fie folosit ca zonă tampon pentru pregătirea afișării*. Rutina pe care o vom numi **DISPSUM** (a nu se confunda cu **DISPNUM**), va transfera numărul din **PRTBUF** în **LEDBUF**, comprimîndu-l totodată pentru a elimina octetul ce conține punctul zecimal.

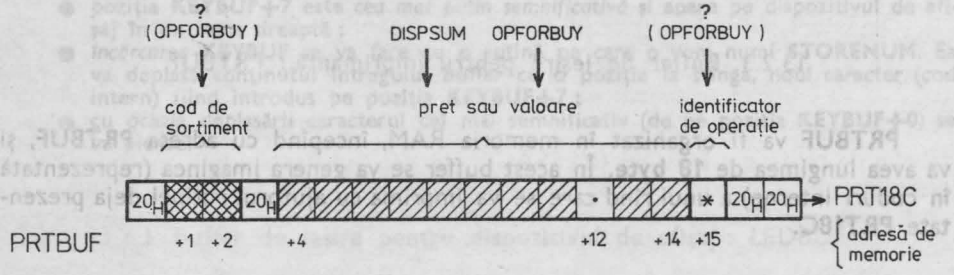


Fig. 13.8. Zone distincte în PRTBUF

Rezultă deci că bufferul **PRTBUF** *este o zonă de manevră multifuncțională*, a cărori intimități se vor identifica în măsura în care proiectul software progresează.

13.2.4. Buffer de editare pentru imprimare : EDBUF

Ţinînd cont de formatul bonurilor de emis (impus prin specificaţia funcţională) precum şi de cele prezentate în paragrafele 13.1.4. (Mesaje în **EPROM**,), vom crea în memoria **RAM** *o zonă de editare pentru pregătirea mesajelor de imprimat*, pe care o vom numi **EDBUF**. Adresa de început a acestei zone de manevră va avea valoarea **EDBUF**, iar lungimea ei va fi egală cu **238** byte.

Această listă de **238 octeţi** va fi organizată pe **14 linii** a cîte **17 octeţi**. Structura unei linii o redăm în fig. 13.9.

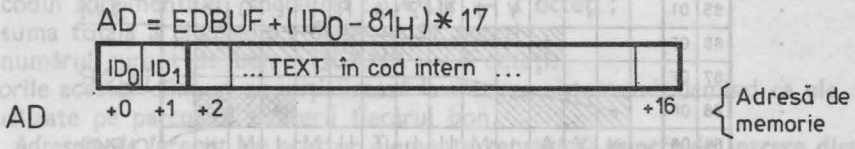


Fig. 13.9. Structura unei linii în EDBUF

În fiecare linie se disting două cîmpuri :

a) **cîmpul de identificare** — constă din **2 octeţi** de identificare ID_0 şi ID_1

b) **cîmpul de text** — constă din **15 octeţi** şi conţine *codurile interne ale caracterelor* ce vor fi imprimate pe un rînd

ID_0 — îl alegem, poate redundant, pentru a-i atribui 2 funcţii :

- avînd bitul cel mai semnificativ înscris (D_7=1) el *va putea servi drept separator* în tabela de date din **EPROM (MESSAGE)**, trăsătură utilizată de rutina **EXPAND** care va genera la trezirea casei de marcat conţinutul iniţial al **EDBUF**
- pe primii patru biţi ai acestui octet (D_0—D_3) vom înscrie un număr de ordine, care poate, ne va fi util.

ID_1 — este conceput ca un *cîmp de selecţii*. Se utilizează primii patru biţi ai săi (D_0—D_3). Ştiind că pe diversele tipuri de bonuri care trebuiesc emise, apar linii comune, ne propunem ca rutina pe care o vom concepe, **PRTTICK**, să primească din rutina apelantă o **mască** (1 octet) **de selecţie**.

Masca de selecţie va avea un singur bit cu valoarea 1, situat pe una din poziţiile D_0 — D_3, restul biţilor fiind zero. Cele 4 măşti preconizate sînt :

- mască pentru bon client : **BUYMASK — 08H**
- mască pentru bon de sortiment : **SORTMASK — 04H**
- mască pentru bon totalizator : **DAYMASK — 02H**
- mască pentru bon de sinteză : **SYNTMASK — 01H**

PRTTICK compară masca de selecţie primită cu cîmpul ID_1 al fiecărei linii din **EDBUF** şi *va imprima linia respectivă dacă bitul corespunzător din* ID_1 *este egal cu cel solicitat prin mască*. Astfel se poate concepe destul de elegant distribuirea diverselor mesaje, comasate iniţial în **EDBUF**, pe diferite tipuri de bon.

Ex. : Bonul client normal va avea masca **08H**, care va determina imprimarea liniilor nr. : **1, 6, 7, 8, 9, 10, 11, 12, 13, 14,**

În figura 13.10. redăm structura generală a bufferului de editare **EDBUF** şi valorile cîmpurilor de selecţie a fiecărei linii.

Aşa cum se vede în fig. 13.10., *în* cîmpul de texte al **EDBUF** *apar texte* care *se copiază* din **EPROM** (prin rutina **EXPAND**) dar *şi valori numerice* (reprezentate în cod intern) care *se generează* în timpul funcţionării casei de marcat.

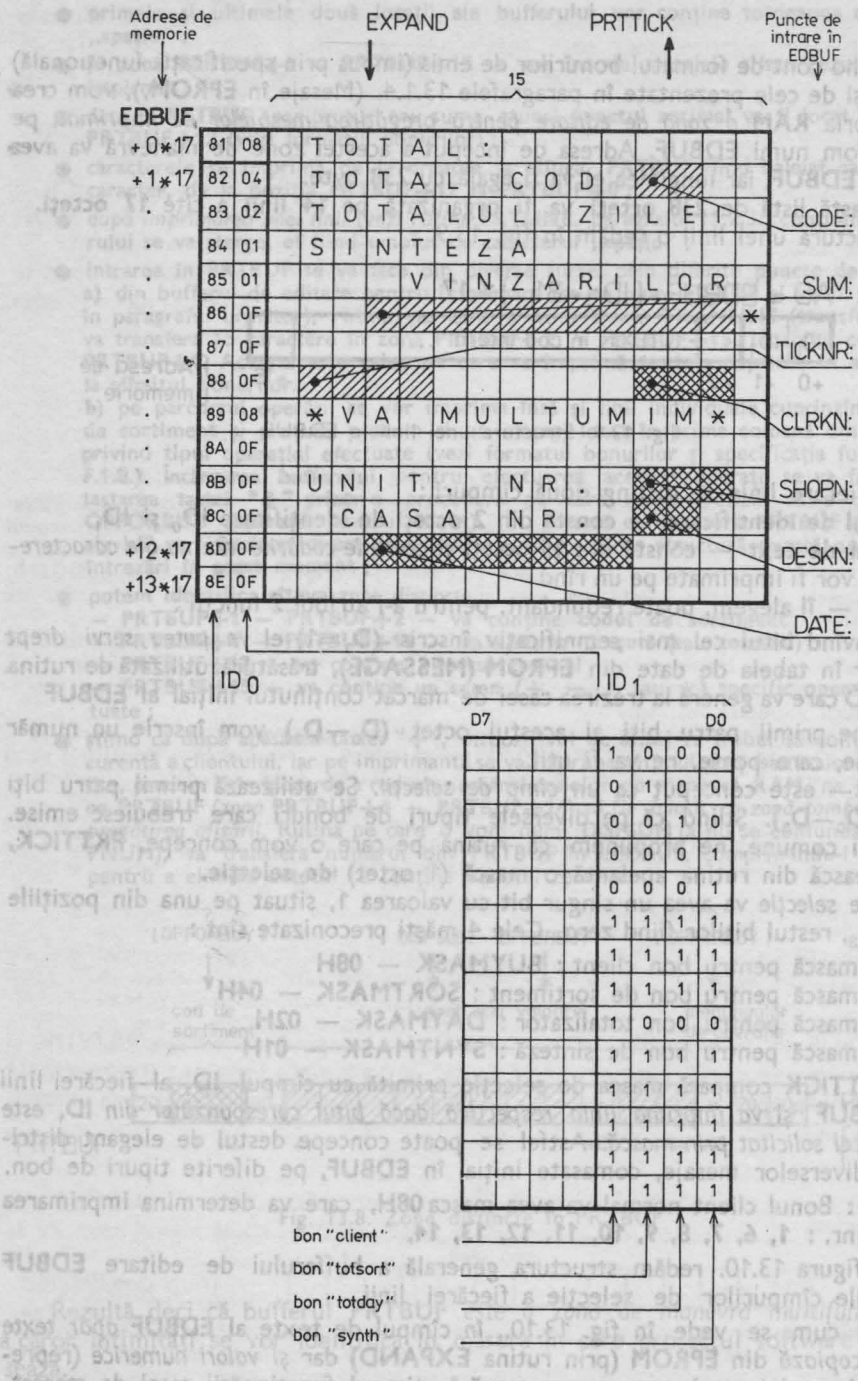


Fig. 13.10. Buffer de editare pentru imprimare : EDBUF

Aceste din urmă elemente le grupăm în *constante* și *variabile*. **Constantele și cîmpurile lor** aferente sînt :

- numărul de identificare al casierului : **CLRKN** — **3** octeți ;
- numărul de identificare al casei : **DESKN** — **2** octeți ;
- numărul de identificare al magazinului : **SHOPN** — **3** octeți ;
- data curentă : **DATE** — **8** octeți.

Aceste cîmpuri se înscriu la programarea sesiunii de lucru conform specificației funcționale **F.2.1.** Ele rămîn nemodificate pînă la nouă reprogramare.

Variabilele și cîmpurile lor aferente cuprinse în **EDBUF** vor fi :

- codul sortimentului specificat : **CODE** — **2** octeți ;
- suma totală a clientului : **SUM** — **11** octeți ;
- numărul curent de bon : **TICKNR** — **4** octeți.

Valorile acestor cîmpuri se inițializează la trezirea sistemului, urmînd ca ele să fie actualizate pe parcursul emiterii fiecărui bon.

Adresele de început ale acestor elemente *reprezintă* **puncte de intrare distincte în EDBUF.** *Lor li se vor atribui nume simbolice*, identice cu numele variabilelor și se vor referi relativ față de adresa de început a zonei **(EDBUF)**, conferind astfel programului un plus de flexibilitate : ele *nu vor trebui modificate dacă într-un* nou proiect *memoria* **RAM** (sau **EDBUF**) *se va dispune într-o altă zonă*. În acest caz se va indica doar noua adresă de început a bufferului.

CODE : = EDBUF + 2 * 17 — 4
SUM. : = EDBUF + 5 * 17 + 5
TICKNR: = EDBUF + 7 * 17 + 3
CLRKN : = TICKNR + 10
SHOPN : = EDBUF + 11 * 17 — 4
DESKN : = EDBUF + 12 * 17 — 4
DATE : = EDBUF + 12 * 17 + 5

Aceste operații se regăsesc în listingul din Cap. 17., pag. 1—1.

13.2.5. Tabela de distribuire a parametrilor de stare : SETUPTAB

Această zonă nu reprezintă de fapt o zonă de manevră, ci *este*, mai exact spus, *o tabelă de distribuire* a unor manevre. În procesul de programare a casei de marcat (specificația funcțională **F.2.1.**), este prevăzută introducerea de la tastatură a parametrilor de stare a casei de marcat : numărul casierului, casei, a magazinului și data curentă. În vederea implementării acestei trăsături vom elabora o rutină pe care o vom numi **SETUP.** Știm că *structura datelor* ce urmează a fi *citite* prin această procedură *este eterogenă* : unele sînt numere „pure" (numerele de identificare) iar altele comportă și prezența punctului zecimal (data calendaristică), el servind ca separator între zi și lună, respectiv lună și an. Dacă dorim să efectuăm și o analiză sintactică a datelor introduse de operator, atunci cele a căror structură diferă, vor trebui tratate diferențiat.

Dacă intrarea va trebui tratată diferențiat, atunci *ne propunem să tratăm măcar ieșirea în mod unitar*. De aceea vom constitui în **EPROM** o tabelă de distribuție **SETUPTAB** care conține adresele de distribuție a parametrilor de stare, și lungimea cîmpului care va fi transferat (lungimea fiecărui parametru, exprimată în octeți). Vom elabora și o rutină care va fi apelată din **SETUP** și va transfera

datele, deja validate din **KEYBUF** în zonele lor dedicate din **EDBUF**. Să numim această rutină **TRANSPAR** (transfer de parametri). Avînd informația structurată conform **SETUPTAB** elaborarea rutinei **TRANSPAR** nu va crea probleme.

Folosind pseudoinstrucțiunile asamblului **M80**, vom defini **SETUPTAB** după cum urmează :

```
SETUPTAB :  DW    SHOPN     ; adresa număr unitate
            DB    SHOPNL    ; lungime număr unitate
            DW    DESKN     ; adresa număr casă
            DB    DESKNL    ; lungime număr casă
            DW    DATE      ; adresa data curentă
            DB    DATEL     ; lungimea cîmpului rezervat
                            ; pentru data curentă
            DW    CLRKN     ; adresa număr casier
            DB    CLRKNL    ; lungime număr casier
```

Ordinea dispunerii celor 4 cîmpuri din **SETUPTAB** a fost impusă de succesiunea specificată în **F.2.1.**

În fig. 13.11. redăm structura **SETUPTAB.**

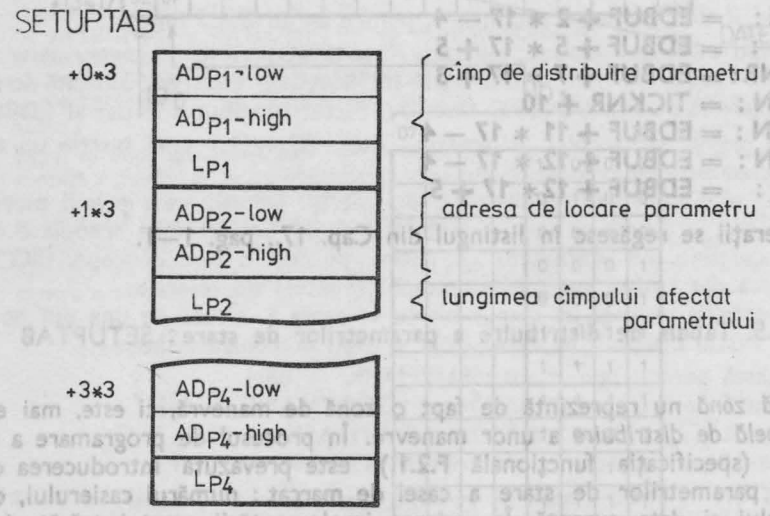


Fig. 3.11. Structura tabelei de distribuire a parametrilor de stare : SETUPTAB

13.3. Regiștri RAM

Știm deja că *sumele* pe care le vom vehicula cu programele casei de marcat *nu pot fi reprezentate* nicidecum *pe unul sau doi octeți.* Forma de reprezentare internă a numerelor (prețuri) asupra cărora va trebui să efectuăm operații aritmetice o vom alege cea **BCD, cu punct zecimal fix.** Suma cea mai mare care ur-

mează să fie vehiculată de această casă, are lungimea de **10 cifre semnificative**, din care 2 sînt rezervate pentru partea zecimală.

Vom încerca să definim rutinele noastre de aritmetică **BCD cu virgulă fixă**, într-o manieră asemănătoare cu cea a microprocesorului. Microprocesorul **Z80** execută toate operațiile aritmetice între 2 numere (binare cu 8 bit) astfel încît unul din parametri va fi locat într-un registru dedicat **(A)**, iar celălalt într-un alt registru. Rezultatul este generat în același registru **A**, suprascriindu-se peste vechea valoare din **A**.

Ținînd cont de faptul că numerele vehiculate de casa de marcat sînt reprezentate pe **5 octeți**, este clar că ele nu vor putea fi locate în regiștri interni ai microprocesorului. De aceea *vom rezerva zone dedicate în memoria* **RAM**, *pe care programele noastre le vor înzestra cu funcții asemănătoare regiștrilor interni ai microprocesorului*. De aceea aceste zone le vom numi **regiștri RAM**.

13.3.1. Registrul datelor în format BCD extins : EBCD

Numerele introduse de la tastatura casei de marcat se regăsesc în **KEYBUF**. Reprezentarea lor poartă următoarele caracteristici :

- *fiecare cifră* este înscrisă într-un octet ;
- *punctul zecimal* apare pe bitul D_7 al cifrei după care a fost introdus
- *numărul* poate fi *nenormalizat*, adică să conțină mai mult decît 2 cifre zecimale, sau nici una ;
- *zerourile nesemnificative* sînt substituite cu blancuri (caracter spațiu).

Pentru programele de aritmetică ale casei de marcat, aceste numere trebuie transpuse într-un format unitar caracterizat prin :

- *lungimea* numărului este unică : 5 octeți ;
- în fiecare octet apar *2 cifre BCD* ;
- *punctul zecimal* nu apare explicit, ci el va fi considerat ca existent între octetul cel mai puțin semnificativ și următorul ;
- *zerourile nesemnificative* trebuie să fie într-adevăr **"0"**.

Transpunerea o vom efectua în două etape :

1. Numărul original se prelucrează și se aduce într-un format concretizat prin :
   - *fiecare cifră* ocupă un octet ;
   - *cifrele zecimale* excedentare sînt eliminate ;
   - *punctul zecimal* nu apare explicit ;
   - *zerourile nesemnificative* sînt într-adevăr "0".

   Acest format îl numim format **BCD extins**.
2. Numerele reprezentate în format **BCD extins** sînt convertite în numere **BCD**, reprezentate conform specificației de mai sus.

Pentru crearea numerelor în format **BCD extins** vom rezerva o zonă tampon în memoria **RAM**, pe care o vom numi registru **EBCD**.

În fig. 13.12. redăm structura registrului **EBCD**, care este plasat în memoria **RAM**, începînd cu adresa **EBCD** și are lungimea de **10 octeți**.

Distingem două cîmpuri :

— **EBCD+0 — EBCD+7** — conține partea întreagă a numărului ;
— **EBCD+8 — EBCD+9** — conține partea zecimală a numărului.

(EBCD+0 este cifra cea mai semnificativă a numărului).

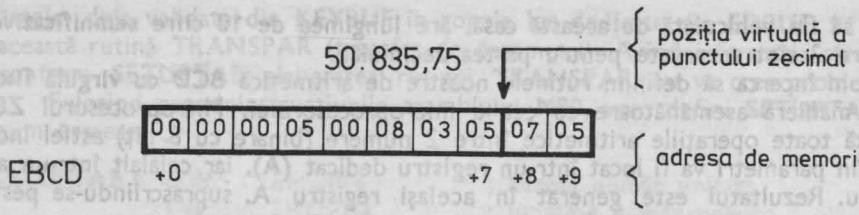


Fig. 13.12. Registrul datelor în format BCD extins : EBCD

Structura unui octet din EBCD este :

— D_7 — D_4 — nesemnificativ (0)
— D_3 — D_0 — cifra zecimală în cod BCD

13.3.2. Regiștrii de aritmetică BCD

Definim : Toate operațiile aritmetice vor avea forma :

TMPBCD : = TMPBCD OP DATBCD

Cei doi regiștri **BCD** vor avea o structură identică, caracterizată prin :

- *lungimea registrului* este de 5 octeți ;
- *partea zecimală* apare în octetul din extrema dreaptă, pe octetul cu cea mai mare adresă de memorie ;
- *partea întreagă* apare în primii 4 octeți ;
- *cifra cea mai semnificativă* apare pe biții D_7 — D_4 și celulei **xBCD+0.** (unde **x** poate fi **TMP** sau **DAT**)

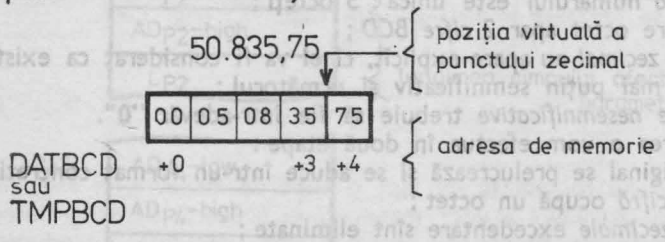


Fig. 13.13. Structura regiștrilor aritmeticii BCD

Preconizăm ca în **DATBCD** să fie depuse numerele introduse de la tastatură. În **TMPBCD** se va depune celălalt operand al unei operații aritmetice. El servește și drept registru rezultat, avînd o funcție numită în general : *acumulator*.

13.3.3. Regiștri BCD de uz general, în memorie

Așa cum rezultă din specificația funcțională a casei de marcat, aceasta va trebui să fie capabilă să și *memoreze* anumite valori : *suma curentă a clientului, totalul vînzărilor* (suma curentă din casă), *sumele vînzărilor pe cele* **100 clase** *de sortimente.*

Aceste valori se vor stoca în forma cea mai comprimată : **BCD**. *Suma totală a clientului* se va crea într-un registru **RAM** pe care-l numim **GUESTBCD**.

Suma totală a vînzărilor din sesiunea respectivă de lucru o vom crea în alt registru **RAM**. Fie numele simbolic al acestui registru : **DAYBCD**.
Structura acestor regiştri este identică cu cea a regiştrilor **DATBCD** respectiv **TMPBCD** (vezi fig. 13.13.).

Pentru memorarea vînzărilor, *defalcată pe dortimente*, va trebui să constituim, *o tabelă în* **RAM**, care va avea lungimea egală cu 100 de regiştri BCD. Cei **500** *de octeţi*, astfel, rezervaţi vor consuma aproape jumătate din memoria RAM disponibilă fizic.

Tabela defalcată a vînzărilor o vom loca la adresa **SORTTOT**. Ea va avea 100 **intrări** corespunzătoare celor 100 de coduri de sortiment.

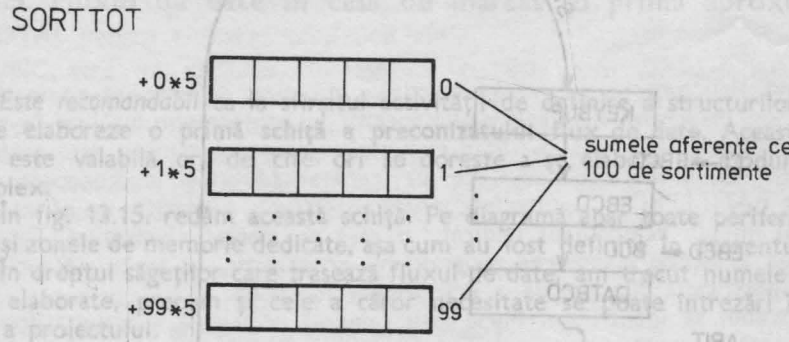


Fig. 13.14. Structura tabelei defalcate a vînzărilor.

Pentru a regăsi totalul aferent unui cod de sortiment dat vom aplica relaţia :

$$ADSORT_i := SORTTOT + 5 * i$$

unde $ADSORT_i$ este adresa de început a cîmpului afectat sortimentului cu codul de sortiment i.

13.4. Parametri şi variabile

Acest paragraf ar trebui să poate titlul „**Diverse**". În faza actuală a proiectului *nu întrezărim decît* **2** elemente :

1. Celula martor pentru portul de ieşire **SYSOUT**, despre rolul căreia s-a vorbit în paragraful 11.2.2.

2. Indicatorul tipului de trezire, cel pe baza căruia se va decide dacă secvenţa de trezire pe care o execută microprocesorul este o **trezire caldă**, sau una **rece**. Trebuie să distingem cazul în care microprocesorul execută o *primă secvenţă* de trezire, *la începutul sesiunii de lucru*, de situaţia în care *repornirea se face la reapariţia tensiunii de alimentare*. În *primul caz* se vor executa o serie de *iniţiali-*

zări iar în *cel de-al doilea activitatea* trebuie *continuată* din punctul în care ea a fost abandonată la dispariția tensiunii de alimentare.

Pentru a putea să luăm o decizie în acest sens, vom compara o secvență din tabela de texte **MESSAGE** din **EPROM**, cu prima linie din **EDBUF**, aflat în memoria **RAM**. Știm că la trezirea sistemului (pornire rece) se va activa rutina **EXPAND** care transpune **MESSAGE** în **EDBUF**. Dacă cele două secvențe sînt

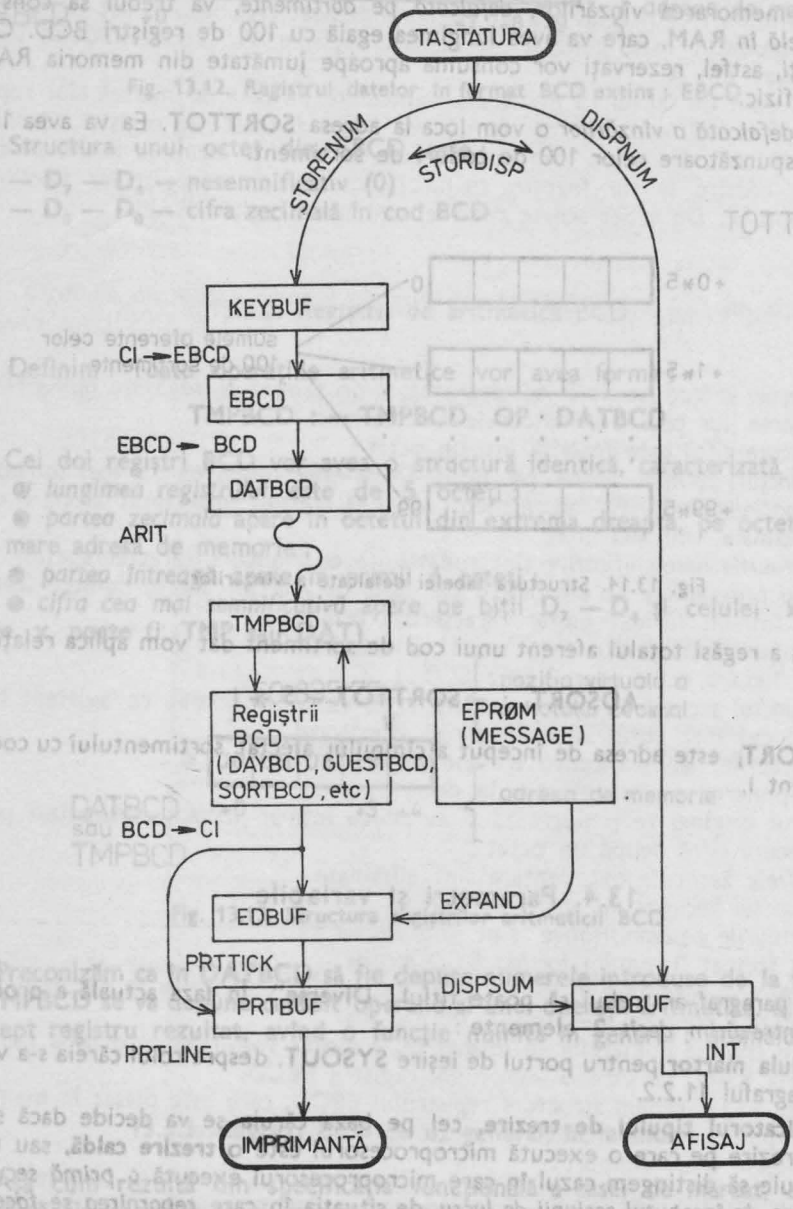


Fig. 13.15. Fluxul de date în casa de marcat (o primă aproximație).

identice, atunci trezirea curentă va fi cea „**caldă**". Acest indicator poate fi bun, deoarece este puțin probabil ca memoria RAM proaspăt alimentată cu tensiune, să se trezească, la adresele respective cu un conținut identic cu cel din **EPROM.**

Primului element i se va rezerva spațiu în memorie : **WITNESS** — se rezervă un octet al cărui conținut va fi identic cu conținutul portului de ieșire **SYSOUT.**

Pentru indicator de trezire nu se va mai rezerva spațiu **RAM,** deoarece el este inclus în **EDBUF.**

În fine, menționăm că este improbabil să fi intuit toate variabilele și datele care pot apărea pe parcursul elaborării proiectului. Nutrind speranța că n-am omis nici una din cele importante, încheiem prezentarea structurilor de date.

13.5. Fluxul de date în casa de marcat (o primă aproximație)

Este recomandabil ca la sfîrșitul activității de definire a structurilor de date, să se elaboreze o primă **schiță** a **preconizatului flux de date.** Această constatare este valabilă ori de cîte ori se dorește a se elabora un modul software complex.

În fig. 13.15. redăm această schiță. Pe diagramă apar toate perifericele precum și zonele de memorie dedicate, așa cum au fost definite în prezentul capitol.

În dreptul săgeților care trasează fluxul de date, am trecut numele rutinelor deja elaborate, precum și cele a căror necesitate se poate întrezări în această fază a proiectului.

Această schiță va fi urmată la sfîrșitul elaborării proiectului software, în mod obligatoriu de o **schiță finală** care va însoți documentația programului.

14
IMPLEMENTAREA PROGRAMULUI

Avînd structura hardware şi structurile de date definite, se poate demara acţiunea de elaborare a **software**-ului specific al echipamentului luat în studiu. Pachetul de programe pe care-l vom elabora în prezentul capitol, este acela care va transforma microcalculatorul de uz general (dotat cu microprocesor Z80, memorie RAM, EPROM şi interfeţe pentru tastatură, dispozitiv de afişaj şi imprimantă) într-un echipament dedicat : o casă de marcat electronică. De aceea, înainte de a începe elaborarea programului propriu-zis, va trebui să recitim cu atenţie **specificaţia funcţională** a echipamentului (Cap. 11.).

Va trebui să identificăm bucla (sau ramura) principală a activităţilor casei. Doar în acel moment se va declanşa elaborarea software-ului : de la esenţă spre detalii.

Pe parcursul întregului capitol vom încerca să respectăm cît mai exact recomandările făcute în Cap. 10., privind succesiunea de elaborare şi de implementare a programelor : vom începe cu modulele ierarhic superioare, coborînd treptat către cele mai simple. Vom lucra permanent cu atenţia distribuită pentru a putea cuprinde problema în totalitatea ei. De aceea nu ne vom hazarda în a elabora o ramură oarecare a programului pînă la ultimul detaliu, înainte de a fi scris modulele ierarhic superioare ale tuturor ramurilor.

Succesiunea de apariţie a paragrafelor din acest capitol este însăşi o ierarhizare a rutinelor. Credem că acest lucru rezultă şi din numele paragrafelor :

14.1. Bucla principală
14.2. Programe de prelucrare
14.3. Aritmetică BCD cu virgulă fixă
14.4. Analiza sintactică şi conversii de coduri
14.5. Vehiculare de date

În ceea ce priveşte metoda generală de lucru precizăm : la început vom descrie algoritmul rutinei studiate, atît prin organigramă, cît şi în limbajul de nivel înalt propus. Cele două aproximaţii grosiere vor fi urmate de prezentarea programului în limbaj de asamblare. După ce familiarizarea cu limbajul descriptiv de nivel înalt se va fi făcut, vom renunţa treptat la organigrame. În măsura în care lucrarea avansează, coborînd spre nivele ierarhice inferioare, odată cu simplificarea problemelor vom renunţa şi la descrierile în limbaj de nivel înalt, păstrînd doar de la caz la caz, cîte o secvenţă semnificativă de program, scris în limbaj de asamblare, şi sau schiţe menite să exemplifice tehnica de implementare adoptată.

14.1. Bucla principală : MAINLOOP

Din analiza specificației funcționale, bucla principală a programului se cristalizează în jurul **„repausului interbon"**. Acesta este punctul la care se ajunge după execuția secvențelor de trezire (**F.1.0.**). Din acest punct se lansează activitățile și tot în acest punct se va reveni după executarea oricăreia din funcțiile specifice ale casei de marcat (**F.1.1.**).

Starea de repaus interbon așteaptă tastarea unei taste, și ea va fi abandonată în momentul în care apare o tastă semnificativă :

- **cifră** sau **punct** pentru introducerea unui preț ;
- **TOTAL** pentru emiterea unui bon vid ;
- **FUNC**, care va declanșa, funcție de tastările care urmează, una din cele **7 acțiuni programabile** : introducerea unui **preț cu cod de sortiment (F.1.13.)**, introducerea unui **preț de ambalaj (F.1.15.)**, **programarea sesiunii de lucru (F.2.1.)**, emiterea unui **bon de anulare (F.2.2.)**, generarea **totalului de sortiment (F.2.3.)**, generarea **totalului de vînzări (F.2.4.)**. și generarea **sintezei vînzărilor (F.2.5.)**. Celelalte taste (**CLEAR**, $+$, $*$) sînt fără semnificație, dacă ele sînt tastate în repausul interbon.

Pe baza acestor considerente constatăm că vor fi două activități majore care se vor putea lansa din repausul interbon :

- introducerea și prelucrarea unui preț **(bon)**
- prelucrarea (procesarea) tastei **FUNC**

Astfel se poate concepe organigrama buclei principale : MAINLOOP (vezi fig. 14.1).

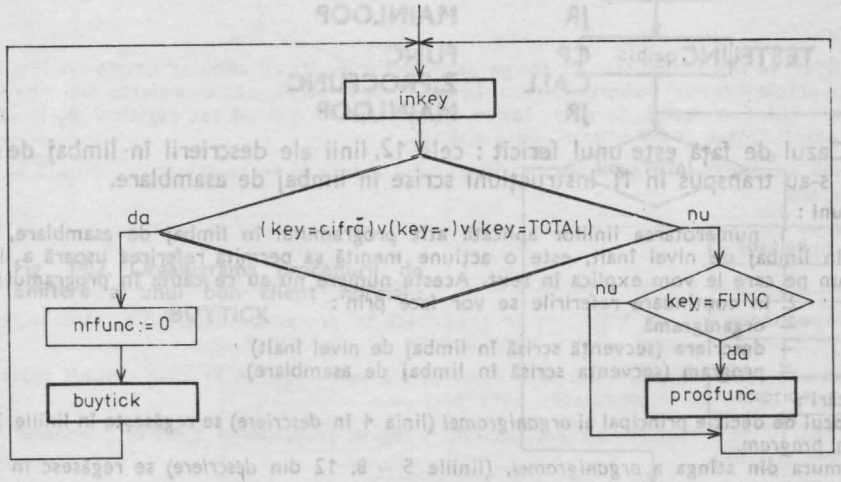


Fig. 14.1. Organigrama buclei principale : MAINLOOP

Sintetizînd : **cifrele, punctul** zecimal și **TOTAL** declanșează procedura legată de emiterea unui bon **(BUYTICK)**, iar tasta **FUNC** va determina activarea unei proceduri de prelucrare **(PROCFUNC)**.

nrfunc — este contorul tastărilor succesive a tastei **FUNC**

Descrierea aceleiași organigrame în limbaj de nivel înalt este :

```
1  repeat
2      begin
3          inkey
4          if   (key=cifră)V(key=punct)V(key=TOTAL)
5          then
6                   begin
7                             nrfunc:=0
8                             buytick
9                   end
10         else
11         if key = FUNC then procfunc
12     end
```

Transpunînd acest program în limbaj de asamblare obținem :

```
1   MAINLOOP :   CALL    INKEY
2                CP      NRORPT
3                JR      C,SIMPBUY
4                CP      TOTAL
5                JR      NZ,TESTFUNC
6   SIMPBUY :    LD      C,0
7                CALL    BUYTICK
8                JR      MAINLOOP
9   TESTFUNC :   CP      FUNC
10               CALL    Z,PROCFUNC
11               JR      MAINLOOP
```

Cazul de față este unul fericit : cele 12, linii ale descrierii în limbaj de nivel înalt, s-au transpus în 11 instrucțiuni scrise în limbaj de asamblare.

Mențiuni :

1. numerotarea liniilor aplicată atît programului în limbaj de asamblare, cît și celui în limbaj de nivel înalt, este o acțiune menită să permită referirea ușoară a liniilor program pe care le vom explica în text. Aceste numere nu au ce căuta în programul sursă.
2. în continuare referirile se vor face prin :
   - **organigramă**
   - **descriere** (secvență scrisă în limbaj de nivel înalt)
   - **program** (secvența scrisă în limbaj de asamblare).

Adnotări :

- **Blocul de decizie** principal al *organigramei* (linia 4 în *descriere*) se regăsește în liniile 2 — 5 din *program*.
- **Ramura din stînga** a *organigramei*, (liniile 5 — 8, 12 din *descriere*) se regăsesc în liniile 6 — 8 ale *programului*.
- **Ramura din dreapta** *organigramei* (liniile 10—12 din *descriere*) se regăsesc în liniile 9 — 11 ale *programului*.
- Pentru delimitarea **cifrelor** și a **punctului** de restul tastelor, am folosit constanta **NRORPT** = **0BH** (linia 2 din *program*), plecînd de la cunoștința codurilor interne ale caracterelor : cifrele și punctul zecimal au coduri mai mici (00—0A$_H$) decît **NRORPT**. Dacă în urma comparației numărul rezultat este negativ (*Cy*=1), atunci tasta apărată a fost cifră sau punct.
- Ca **numărător al tastărilor succesive** ale tastei **FUNC**, vom folosi registrul **C** (linia 7 din *descriere*, respectiv linia 6 din *program*), element de reținut.

Bucla principală se regăseşte în listingul din Cap. 17., pag. 1—9.

Trecem în continuare la elaborarea grosieră a celor două proceduri folosite în **MAINLOOP : BUYTICK şi PROCFUNC.**

14.1.1. Bonul client normal : BUYTICK

Rememorînd specificaţia tehnică, rezultă că activităţile legate de emiterea unui **bon client normal** se pot sintetiza în felul următor :

- ea este declanşată prin introducerea unui preţ ;
- pe parcursul ei se pot introduce oricîte preţuri separate prin tasta "+" ; între două preţuri se stă în repausul interpreţ **(F.1.9.)** ;
- procedura se termină prin apăsarea tastei **TOTAL**, care va declanşa emiterea fizică (imprimarea) a bonului client normal **(F.1.10.)** ;
- după terminarea secvenţei se revine în repausul interbon **(MAINLOOP) (F.1.11.)**.

Rezultă deci, că bucla principală a procedurii **BUYTICK** va fi cea care porneşte şi se întoarce în **repausul interpreţ.**

În fig. 14.2. redăm organigrama care se poate constitui pe baza celor enunţate mai sus.

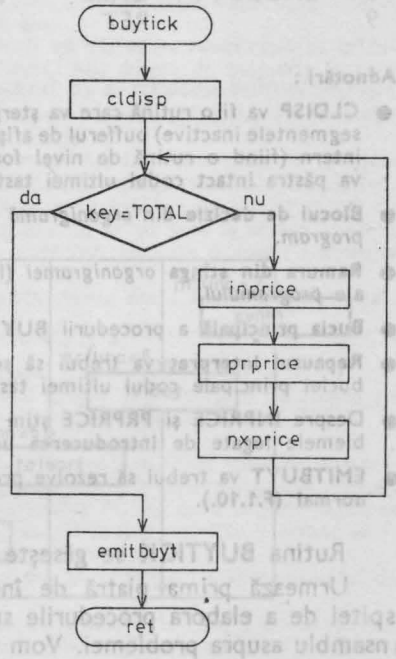


Fig. 14.2. Organigrama procedurii de emitere a unui bon client normal : BUYTICK

Redăm acțiunile din organigramă și într-o descriere în limbaj de nivel înalt.

```
    procedure buytick
1    begin
2         cldisp {ștergere afișaj}
3         while key ≠ TOTAL do
4              begin
5                   inprice {se citește un preț de la tastatură}
6                   prprice {se prelucrează prețul citit}
7                   nxprice {începe citirea unui preț nou}
8              end
9         emitbuyt {se emite bonul client normal}
10   end
```

Rezultă programul scris în limbaj de asamblare :

```
1   BUYTICK :  CALL     CLDISP
2   BTESTTOT : CP       TOTAL
3              JR       Z,ENDBUYT
4              CALL     INPRICE
5              CALL     PRPRICE
6              CALL     NXPRICE
7              JR       BTESTTOT
8   ENDBUYT :  CALL     EMITBUYT
9              RET
```

Adnotări :

- **CLDISP** va fi o rutină care va șterge dispozitivul de afișaj, adică va umple cu FF_H (toate segmentele inactive) bufferul de afișaj **LEDBUF**. În principiu, ea nu va afecta nici un registru intern (fiind o rutină de nivel foarte scăzut) dar în nici un caz registrul **A**, în care se va păstra intact codul ultimei taste apăsate.
- **Blocul de decizie** din *organigramă* (linia 3 din *descriere*) se regăsește în liniile 2 — 3 din *program*.
- **Ramura din stînga** *organigramei* (liniile 9—10 din *descriere*) se regăsesc în liniile 8 — 9 ale *programului*.
- **Bucla principală** a procedurii **BUYTICK** se regăsește în liniile 2—7 ale *programului*.
- **Repausul interpreț** va trebui să se regăsească în procedura **NXPRICE**, care va restitui buclei principale codul ultimei taste apăsate, în registrul **A**.
- Despre **INPRICE** și **PRPRICE** știm doar, că ele vor trebui să rezolve absolut toate problemele legate de **introducerea unui preț**.
- **EMITBUYT** va trebui să rezolve problemele legate de **emiterea** efectivă a **bonului client normal (F.1.10.)**.

Rutina **BUYTICK** se găsește în listingul din Cap. 17, pag. 1—10.

Urmează prima piatră de încercare a începătorilor : va trebui să rezistăm ispitei de a elabora procedurile sus menționate, pentru a nu pierde viziunea de ansamblu asupra problemei. Vom reveni la **MAINLOOP**, pentru a-i trata cealaltă procedură : **PROCFUNC**.

14.1.2. Procesarea tastei FUNC : PROCFUNC

Tasta **FUNC** fiind multifuncțională, numărul de tastări succesive ale ei va trebui să declanșeze activități diferite ale casei de marcat. Aceste activități sînt impuse prin specificația funcțională (**F.2.x.** respectiv **F.1.13.** și **F.1.15.**).

■ Procedeul de a declanșa activități complet diferite prin tastarea repetată a uneia și aceleiași taste, este o operație destul de exigentă pentru operator. De aceea nu se va uita că el poate greși, prevederea măsurilor, de ergonomie formulate în specificația funcțională (indicarea numărului de tastări succesive pe afișaj și prevederea posibilității de a anula o comandă înainte de a o executa efectiv, (**F2.0.**) fiind obligatorii.

În fig. 14.3. redăm organigrama procedurii **PROCFUNC**.

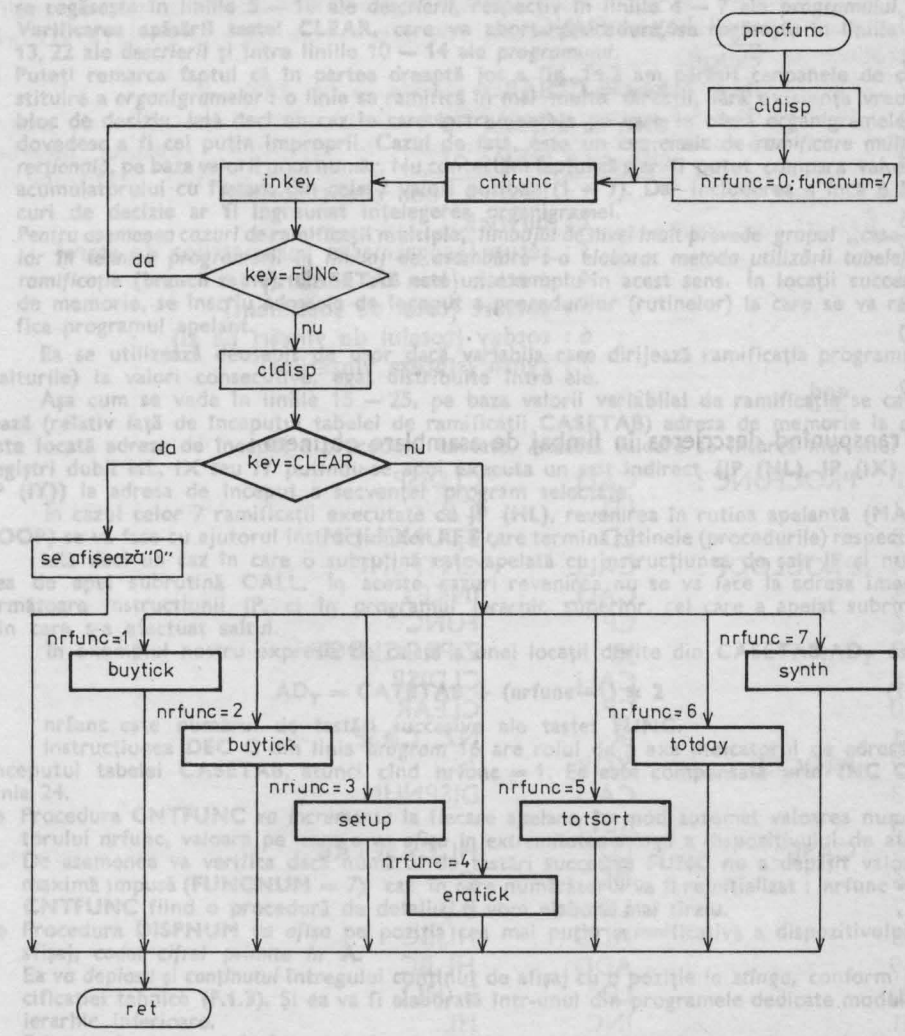


Fig. 14.3. Organigrama procedurii de prelucrare a tastărilor succesive FUNC : PROCFUNC

Elaborăm descrierea în limbaj de nivel înalt :

```
procedure procfunc
  begin
    cldisp {ştergere afişaj}
    nrfunc :=0
    funcnum:=7
    repeat
      begin
        cntfunc {contorizare FUNC şi afişare}
        inkey
      end
    until key ≠ FUNC
    cldisp
    if    key = CLEAR
    then se afişează "0"
    else case nrfunc of
           1 : buytick {bon client normal}
           2 : buytick
           3 : setup {programarea parametrilor de stare}
           4 : eratick {bon de anulare}
           5 : totsort {total de sortiment}
           6 : totday {totalul de vînzări pe zi}
           7 : synth {sinteza vînzărilor}
  end
```

Transpunînd descrierea în limbaj de asamblare obţinem :

```
PROCFUNC :  CALL   CLDISP
            LD     C,0
            LD     B,FUNCNUM
PROCLOOP :  CALL   CNTFUNC
            CALL   INKEY
            CP     FUNC
            JR     Z,PROCLOOP
            CALL   CLDISP
            CP     CLEAR
            JR     NZ, CASE
PROCCL :    XOR    A
            CALL   DISPNUM
            RET
CASE :      LD     HL,CASETAB
            DEC    C
            LD     B,0
            ADD    HL,BC
            ADD    HL,BC
            LD     E,(HL)
            INC    HL
            LD     D,(HL)
            EX     DE,HL
```

```
24          INC   C
25          JP    (HL)
26  CASETAB : DW    BUYTICK
27          DW    BUYTICK
28          DW    SETUP
29          DW    ERATICK
30          DW    TOTSORT
31          DW    TOTDAY
32          DW    SYNTH
```

Adnotări :

- **Bucla de contorizare** a tastărilor (cea care include procedura cntfunc) din *organigramă*, se regăsește în liniile 5 — 10 ale *descrierii*, respectiv în liniile 4 — 7 ale *programului*.
- **Verificarea apăsării tastei CLEAR**, care va aborta procedura, se regăsește în liniile 12, 13, 22 ale *descrierii* și între liniile 10 — 14 ale *programului*.
- Puteți remarca faptul că în partea dreaptă jos a fig. 14.3 am părăsit canoanele de constituire a *organigramelor* : o linie se ramifică în mai multe direcții, fără existența vreunui bloc de decizie. Iată deci un caz în care instrumentele pe care le oferă organigramele se dovedesc a fi cel puțin improprii. Cazul de față, este un caz clasic de *ramificare multidirecțională*, pe baza valorii unui număr. Nu contestăm faptul că s-ar fi putut compara valoarea acumulatorului cu fiecare din cele 7 valori posibile (1 — 7). Dar includerea a încă 6 blocuri de decizie ar fi îngreunat înțelegerea organigramei.
  Pentru asemenea cazuri de **ramificații multiple**, *limbajul de nivel înalt prevede grupul* **„case-of"** *iar în tehnica programării în limbaj de asamblare s-a elaborat metoda utilizării tabelei de ramificație* **(branch table). CASETAB** este un exemplu în acest sens. În locații succesive de memorie, se înscriu adresele de început a procedurilor (rutinelor) la care se va ramifica programul apelant.

Ea se utilizează deosebit de ușor dacă variabila care dirijează ramificația programului (salturile) ia valori consecutive, egal distribuite între ele.

Așa cum se vede în liniile 15 — 25, pe baza valorii **variabilei de ramificație** se calculează (relativ față de începutul tabelei de ramificații **CASETAB**) adresa de memorie la care este locată adresa de început a procedurii căutate. Această valoare se încarcă într-unul din regiștri dubli **HL, IX** sau **IY** putîndu-se apoi executa un salt indirect **(JP (HL), JP (IX)** sau **JP (IY))** la adresa de început a secvenței program selectate.

În cazul celor 7 ramificații executate cu **JP (HL)**, revenirea în rutina apelantă **(MAINLOOP)** se va face cu ajutorul instrucțiunilor **RET** care termină rutinele (procedurile) respective.

Iată deci un caz în care o subrutină este apelată cu instrucțiunea de salt **JP** și nu cu cea de apel subrutină **CALL**. În aceste cazuri revenirea nu se va face la adresa imediat următoare instrucțiunii **JP**, ci în programul ierarhic superior, cel care a apelat subrutina din care s-a efectuat saltul.

În exemplul nostru expresia de calcul a unei locații dorite din CASETAB, AD_T este :

$$AD_T = CATETAB + (nrfunc - 1) * 2 \qquad (14.1)$$

nrfunc este numărul de tastări succesive ale tastei **FUNC**.

Instrucțiunea **DEC C** din linia *program* 16 are rolul de a axa indicatorul de adresă pe începutul tabelei **CASETAB**, atunci cînd **nrfunc** = 1. Ea este compensată prin **INC C** în linia 24.

- Procedura **CNTFUNC** *va incrementa* la fiecare apelare, în mod automat valoarea numărătorului **nrfunc**, valoare pe care *o va afișa în extremitatea stîngă* a dispozitivului de afișaj. De asemenea va verifica dacă numărul de tastări succesive **FUNC** nu a depășit valoarea maximă impusă (**FUNCNUM** = 7), caz în care numărătorul va fi reinițializat : **nrfunc** = 1. **CNTFUNC** fiind o procedură de detaliu, o vom elabora mai tîrziu.
- Procedura **DISPNUM** *va afișa* pe poziția cea mai puțin semnificativă a dispozitivului de afișaj, *codul cifrei primite în* **A**.
  Ea *va deplasa și conținutul* întregului conținut de afișaj cu o poziție *la stînga*, conform specificației tehnice **(F.1.3)**. Și ea va fi elaborată într-unul din programele dedicate modulelor ierarhic inferioare.
- Primele două cazuri (**nrfunc** =1 și **nrfunc**=2) ne vor conduce la procedura deja prezentată **BUYTICK**. Este vorba de un preț introdus cu **cod de sortiment** (**nrfunc**=1) sau un **preț de ambalaj** restituit (**nrfunc**=2).

Listingul comentat al rutinei **PROCFUNC** se găsește în Cap. 17, pag. 1—11.

Menținîndu-ne în continuare pe primul nivel ierarhic, vom elabora procedurile dedicate funcțiilor speciale ale casei de marcat.

14.1.3. Programarea sesiunii de lucru : SETUP

Modul de programare a **parametrilor de stare** ai sesiunii de lucru *(magazin, număr casă, număr casier, dată)* este impus prin specificația tehnică **F.2.1.**

- Procedura **SETUP** va fi declanșată după **trei** tastări succesive ale tastei **FUNC.**
- Pe parcursul programării cei 4 *parametri vor fi separați prin* apăsarea tastei **TOTAL.** În fiecare fază de introducere *tasta* **CLEAR** *va fi activă* permițînd ștergerea unei date eronat introduse.
- Prin natura lor, *datele* care urmează să fie introduse, *ne scutesc de analiza corectitudinii.* Singura măsură de protecție pe care o vom lua este aceea ca, codurile introduse să fie cifră sau punct.

Organigrama procedurii **SETUP** se găsește în fig. 14.4.

După cea de-a 4-a tastare a tasei **TOTAL** (la terminarea procesului de programare) urmează revenirea în bucla principală de așteptare : **repausul interbon (MAINLOOP).** Vom semnala acest eveniment și pe cale acustică, pentru un plus de ergonomie.

Rezultă următoarea descriere în limbaj de nivel înalt :

```
procedure setup
1   begin
2       prog :=0
3       repeat
4           begin
5               repeat
6                   begin
7                       while (key ≠ cifră) ∧ (key ≠ punct) do
8                           begin
9                               inkey
10                              if key=CLEAR then „reinitializare buffere"
11                          end
12                      stordisp {memorare și afișare cifră sau}
13                                {punct}
14                      inkey
15                  end
16              until key = TOTAL
17          transpar {parametrul introdus se transferă la locul lui}
18          clbufdp {reinițializarea bufferelor }
19          prog : = prog+1
20      end
21      until prog = 4
22      se afișează "0"
23      sonerie
    end
```

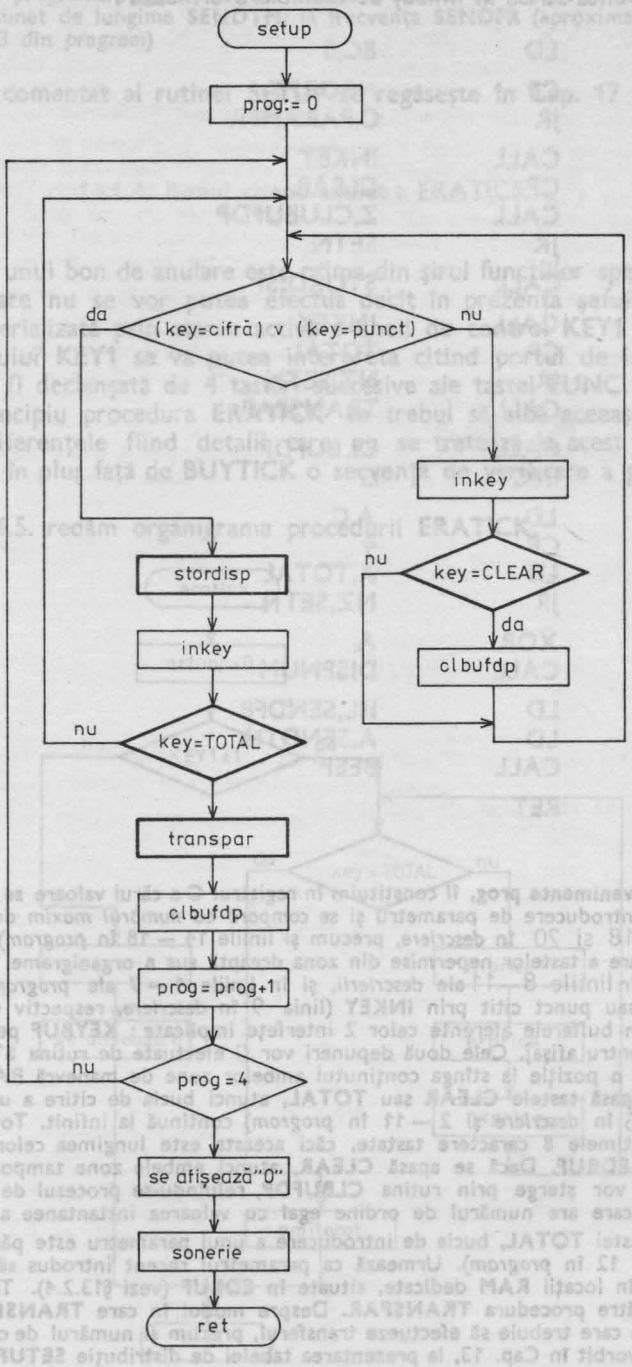


Fig. 14.4. Organigrama procedurii de programare a parametrilor de stare : SETUP

Rutina aferentă scrisă în limbaj de asamblare urmează :

```
1  SETUP :   LD      BC,0
2  SETN :    CP      NRORPT
3            JR      C,PARAMPR
4            CALL    INKEY
5            CP      CLEAR
6            CALL    Z,CLUBUFDP
7            JR      SETN
8  PARAMPR : CALL    STORDISP
9            CALL    INKEY
10           CP      TOTAL
11           JR      NZ,SETN
12           CALL    TRANSPAR
13           CALL    CLBUFDP
14           INC     C
15           LD      A,C
16           CP      4
17           LD      A,TOTAL
18           JR      NZ,SETN
19           XOR     A
20           CALL    DISPNUM
21           LD      HL,SENDFR
22           LD      A,SENDTN
23           CALL    BEEP
24           RET
```

Adnotări :

- **Contorul de evenimente prog,** îl constituim în registrul **C** a cărui valoare se *incrementează* după fiecare introducere de parametru şi se *compară* cu *numărul maxim* de parametri 4. (vezi liniile 18 şi 20 în *descriere*, precum şi liniile 14 — 18 în *program*).
- Bucla de filtrare a tastelor nepermise din zona dreaptă sus a organigramei din fig. 14.4 se regăseşte în liniile 8—11 ale *descrierii*, şi în liniile 4 — 7 ale *programului*.
- Fiecare cifră sau punct citit prin **INKEY** (linia 9 în *descriere*, respectiv 4 în *program*) este depusă în bufferele aferente celor 2 interfeţe implicate : **KEYBUF** pentru tastatură şi **LEDBUF** pentru afişaj. Cele două depuneri vor fi efectuate de rutina **STORDISP** care va deplasa cu o poziţie la stînga conţinutul ambelor zone de manevră RAM.
- Dacă nu se apasă tastele **CLEAR** sau **TOTAL,** atunci bucla de citire a unui parametru (liniile 5—15 în *descriere* şi 2 — 11 în *program*) continuă la infinit. Totdeauna vor fi disponibile ultimele 8 caractere tastate, căci aceasta este lungimea celor două buffere **KEYBUF** şi **LEDBUF.** Dacă se apasă **CLEAR,** atunci ambele zone tampon **(KEYBUF** şi **LEDBUF)** se vor şterge prin rutina **CLBUFDP,** reluîndu-se procesul de introducere a parametrului care are numărul de ordine egal cu valoarea instantanee a registrului **C.**
- La apăsarea tastei **TOTAL,** bucla de introducere a unui parametru este părăsită (linia 16 în *descriere* şi 12 în *program*). Urmează ca parametrul recent introdus să fie transferat din **KEYBUF** în locaţii **RAM** dedicate, situate în **EDBUF** (vezi §13.2.4). Transferul va fi efectuat de către procedura **TRANSPAR.** Despre modul în care **TRANSPAR** va determina adresa la care trebuie să efectueze transferul, precum şi numărul de octeţi de transferat s-a mai vorbit în Cap. 13, la prezentarea tabelei de distribuţie **SETUPTAB,** şi se va mai vorbi la prezentarea procedurii însăşi. Cert este că ea **(TRANSPAR)** va folosi registrul **C** ca **identificator al parametrului** de transferat.

● *Procedura* de programare a parametrilor de stare a casei de marcat *se termină* prin emiterea unui sunet de lungime **SENDTM**, la frecvența **SENDFR** (aproximativ 3 kHz), (vezi liniile 21—23 din *program*)

Listingul comentat al rutinei **SETUP** se regăsește în Cap. 17 la pag. 1—14.

14.1.4. Bonul client anulat : ERATICK

Emiterea unui bon de anulare este prima din șirul funcțiilor speciale ale casei de marcat, care nu se vor putea efectua decît în prezența șefului de unitate, prezență materializată prin starea activă a cheii de control **KEY1 (F.2.2.)**. Activarea semnalului **KEY1** se va putea interpreta citind portul de intrare **SYSIN**. Procedura va fi declanșată de 4 tastări succesive ale tastei **FUNC**.

■ În principiu procedura **ERATICK** va trebui să aibă aceeași structură cu **BUYTICK**, diferențele fiind detalii care nu se tratează la acest nivel ierarhic. Ea va conține în plus față de **BUYTICK** o secvență de verificare a prezenței cheii de control.

În fig. 14.5. redăm organigrama procedurii **ERATICK**.

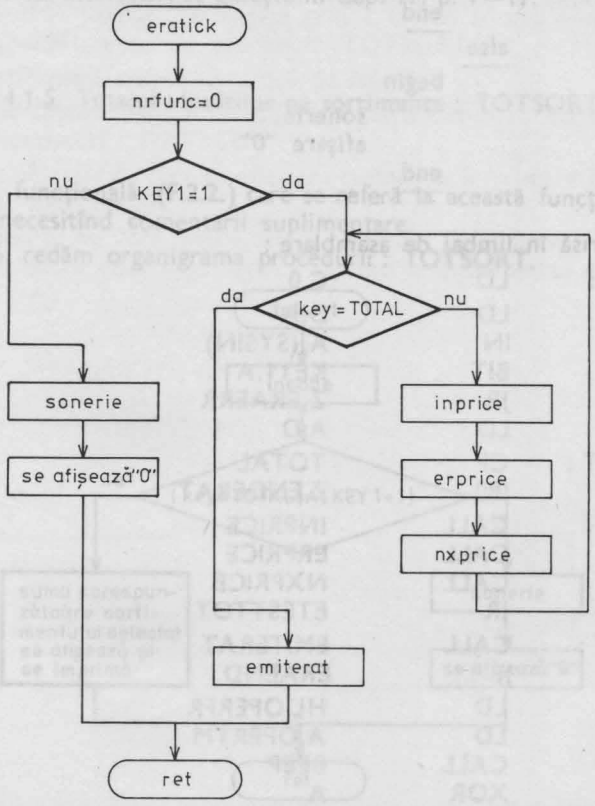


Fig. 14.5. Organigrama procedurii de emitere a unui bon de anulare : ERATICK

● În ramura din stînga a fig. 14.5. se vede că orice încercare de a declanșa procedura **ERATICK**, în absența cheii de control, este rejectată, emițîndu-se totodată un semnal sonor de avertizare.

```
procedure eratick
  begin
      nrfunc : = 0
      if key 1=1
          then
              begin
                  while key ≠ TOTAL do
                      begin
                          inprice {se citește un preț de la
                                   tastatură}
                          erprice {se anulează}
                          nxprice {începe citirea unui nou
                                   preț}
                      end
                  emiterat {se emite bonul de anulare}
              end
          else
              begin
                  sonerie
                  afișare "0"
              end
  end
```

Rutina scrisă în limbaj de asamblare :

```
ERATICK :   LD      C,0
            LD      D,A
            IN      A,(SYSIN)
            BIT     KEY1,A
            JR      Z,ERAERR
            LD      A,D
ETESTTOT :  CP      TOTAL
            JR      Z,ENDERAT
            CALL    INPRICE
            CALL    ERPRICE
            CALL    NXPRICE
            JR      ETESTTOT
ENDERAT :   CALL    EMITERAT
            JR      ERAEND
ERAERR :    LD      HL,OPERFR
            LD      A,OPERTM
            CALL    BEEP
            XOR     A
            CALL    DISPNUM
ERAEND :    RET
```

Adnotări :

- Datorită faptului că procedura **INPRICE** va citi eventual şi preţuri precedate de codul de marfă (deci tastări succesive **FUNC**) numărătorul aferent — **nrfunc** — va trebui să fie iniţializat întocmai ca şi în cazul procedurii **BUYTICK.** (Reţinem faptul că **BUYTICK** primeşte din **MAINLOOP** mereu **nrfunc** : = 0) ; **nrfunc** este contorizat în registrul **C**.
- Constatăm aici că *primul preţ introdus* după cele 4 tastări **FUNC** *nu poate fi precedat de cod de sortiment şi nu poate fi preţ de ambalaj.* Ambele ar trebui să fie precedate de noi tastări **FUNC,** ceea ce ar incrementa **nrfunc,** selectîndu-se astfel o altă funcţie specială a casei de marcat (nu s-a părăsit **PROCFUNC**). De aceea recomandăm operatorului să înceapă emiterea fiecărui bon de anulare introducînd după cele patru tastări **FUNC,** un preţ „0", care va declanşa procedura **ERATICK,** şi nu va afecta sumele memorate.
- Ramura de evitare (lipsă **KEY1**) apare în *descriere* în liniile 14—18, iar în *program* între liniile 15 şi 20.
- Remarcăm faptul că primul argument al instrucţiunii **BIT** din linia *program* 4 este un simbol. Astfel rutina îşi va păstra valabilitatea şi în condiţiile modificării hardware-ului, programatorul va atribui simbolului **KEY1** o altă valoare. La o nouă asamblare aceste valori se vor substitui în toate instrucţiunile care au folosit **KEY1.**
- Procedurile **ERPRICE** şi **EMITERAT** nu vor diferi structural de **PRPRICE** şi **EMITBUYT.** În esenţă ele *vor inversa sensul unor operaţii aritmetice* şi vor *imprima un mesaj special* pe bonul emis **(F.2.2.).**

Listingul rutinei **ERATICK** se găseşte în Cap. 17, p. 1—17.

14.1.5. Totalul vînzărilor pe sortimente : TOTSORT

Specificaţia funcţională **(F.2.2.)** care se referă la această funcţie specială este clară, ea nemainecesitînd comentarii suplimentare.

În fig. 14.6. redăm organigrama procedurii : **TOTSORT.**

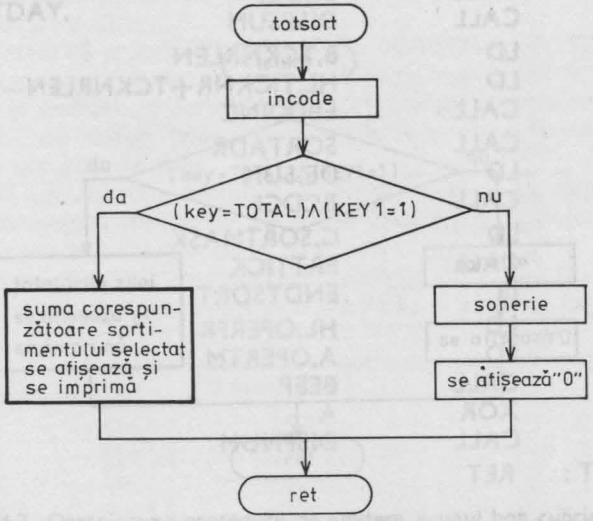


Fig. 14.6. Organigrama procedurii de emitere a unui bon cuprinzînd totalul unui sortiment : TOTSORT

Descrierea în limbaj de nivel înalt a fig. 14.6. este :

```
1  procedure totsort
1      begin
2          incode {se citeşte de la tastatură codul sortimentului dorit}
3          if (key = TOTAL) ∧ (KEY1=1)
4              then
5                  begin
6                  "suma corespunzătoare sortimentului selectat,
                   se imprimă şi se afişează"
7                  end
8              else
9                  begin
10                     sonerie
11                     afişare „0"
12                 end
13     end
```

Descrierea în limbaj de nivel înalt este destul de vagă, în speţă în linia 6. Vom detaila această linie în *programul* scris în limbaj de asamblare.

```
1  TOTSORT :   CALL    INCODE
2              CP      TOTAL
3              JR      ERRTSORT
4              IN      A,(SYSIN)
5              BIT     KEY1,A
6              JR      Z,ERRTSORT
7              CALL    SORTADR
8              LD      DE,PRTBUF+4
9              CALL    BCDCI
10             CALL    DISPSUM
11             LD      B,TCKNRLEN
12             LD      HL,TICKNR+TCKNRLEN-1
13             CALL    EBCDINC
14             CALL    SORTADR
15             LD      DE,SUM
16             CALL    BCDCI
17             LD      C,SORTMASK
18             CALL    PRTTICK
19             JR      ENDTSORT
20 ERRTSORT :  LD      HL,OPERFR
21             LD      A,OPERTM
22             CALL    BEEP
23             XOR     A
24             CALL    DISPNUM
25 ENDTSORT :  RET
```

Adnotări :

- Procedura **INCODE** *va citi un cod de sortiment* format din 2 cifre şi îl va depune în **EDBUF** la intrarea **CODE** (vezi fig. 13.10). Pe parcursul citirii celor două cifre *tasta* **CLEAR** *va fi activă* (vezi specificaţia funcţională **F.1.13** şi **F.1.15**).

- Linia 6 din *descriere* s-a transpus în liniile 7 — 19 din *program*.
- Procedura **SORTADR** *va determina adresa de început a registrului* **BCD** *în* **RAM** (*tabela* **SORTTOT**) care conține totalul vînzărilor aferente codului de sortiment din **CODE**. **SORTADR** va restitui în **HL** adresa de început a registrului căutat. (structura tabelei **SORTTOT** se găsește în §13.3.3).
- Sumele și prețurile sînt memorate în cod **BCD**, ele vor trebui convertite în cod intern. Rutina **BCDCI** *convertește* valoarea **BCD** din registrul **RAM** pontat de **HL** și îl depune într-o zonă adresată de **DE**. (în cazul de față **PRTBUF**).
- Procedura **DISPSUM** *va afișa* un număr din **PRTBUF**. **DISPSUM** elimină și zerourile nesemnificative din stînga numărului.
- Folosind masca de selecție **SORTMASK**, **PRTTICK** va imprima conținutul lui **EDBUF**, selectînd liniile care au bitul de coincidență setat (vezi §13.2.4).
- Procedura **TOTSORT** emite același sunet (caracterizat prin durata **OPERTM** și frecvența **OPERFR**) ca și **ERATICK** în caz de operare greșită :
  — fie că nu este prezentă cheia de control (liniile 4—6 din *program*) ;
  — fie că nu s-a apăsat tasta **TOTAL** după introducerea codului de sortiment selectat.

Listingul comentat al rutinei **TOTSORT** se găsește în Cap. 17. pag. 1—18.

14.1.6. Totalul vînzărilor pe zi : TOTDAY

Procedura acestei funcții, declanșată prin 6 tastări succesive a tastei **FUNC**, diferă prin puține elemente de procedura **TOTSORT** :

- *nu* este necesară citirea unui *cod de marfă* ;
- *nu trebuie căutat* **registrul RAM** care conține suma ; cerută, căci adresa ei este cunoscută : **DAYBCD** ;
- **masca de selecție** cu care se apelează rutina de imprimare bon, **PRTTICK**, este alta : **DAYMASK**.

În aceste condiții nu vom insista asupra modului de elaborare a programului, publicînd în continuare doar organigrama și descrierea în limbaj de nivel înalt a procedurii **TOTDAY**.

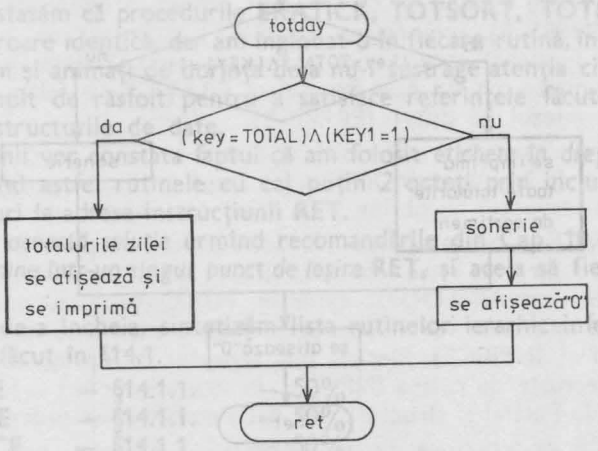


Fig. 14.7. Organigrama procedurii de emitere a unui bon cuprinzînd totalul vînzărilor pe zi : TOTDAY

Descrierea :

```
procedure totday
          begin
                    if (key=TOTAL) ∧ (KEY1=1)
                              then
                                        begin
                                                  „totalul zilei din DAYBCD se imprimă
                                                  și se afișează"
                                        end
                              else
                                        begin
                                                  sonerie
                                                  afișează „0'
                                        end
          end
```

În Cap. 17, pag. 1—19 se găsește listingul comentat al rutinei **TOTDAY.**

14.1.7. Sinteza vînzărilor : SYNTH

Specificată funcțional **(F.2.5.),** această comandă se declanșează prin **7 tastări** succesive **FUNC,** urmată de tasta **TOTAL.**

■ Se vor lista în ordinea crescătoare a codurilor de sortiment toate sumele aferente.

Organigrama procedurii **SYNTH** se găsește în fig. 14.8.

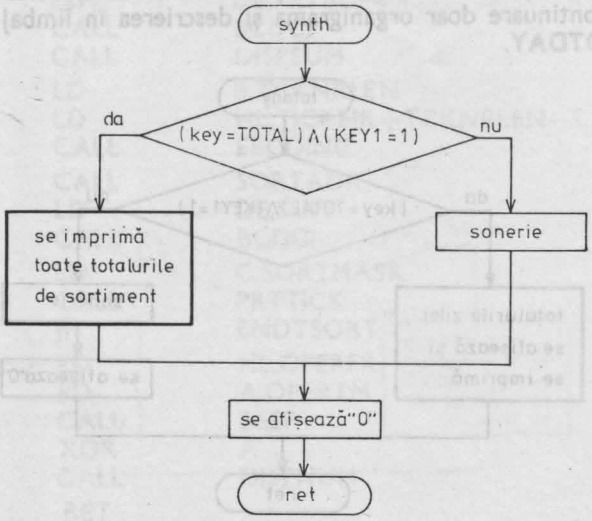


Fig. 14.8. Organigrama procedurii de listare a sintezei vînzărilor SYNTH

Descrierea procedurii în limbaj de nivel înalt este la fel de compactă :

```
procedure synth
        begin
                if (key=TOTAL) ∧ (KEY1=1)
                        then "se imprimă sumele aferente sortimentelor"
                        else „se sună soneria"
                                „se afişează "0""
        end
```

Imprimarea celor 100 de linii de date o lăsăm pe seama rutinei **PRTTICK,** care primind masca de selecţie **SYNTMASK,** va dispune de toate informaţiile necesare pentru a imprima bonul de sinteză. Ea va prelua rînd pe rînd conţinuturile regiştrilor **RAM** din **SORTTOTT** şi le va imprima.

Listingul comentat al acestei proceduri se găseşte în Cap. 17, pag. 1—20.

Prin elaborarea ultimei funcţii majore considerăm **nivelul ierarhic zero al software**-ului casei de marcat epuizat.

Înainte de a aborda următorul nivel vom sintetiza activitatea desfăşurată :

1. Procedurile **nivelului ierarhic zero** reprezintă o implementare fidelă a sarcinilor impuse prin specificaţia funcţională. *Scopul principal urmărit* n-a fost *cel* de-a etala tehnici elevate de programare în limbaj de asamblare, ci *de a prezenta modul în care o specificaţie tehnică trebuie transpusă în program,* şi de a obişnui cititorul cu un stil de muncă : *specificaţie, organigramă* sau *descriere în limbaj de nivel înalt, program.*

2. *Aceste proceduri nu rezolvă nimic concret,* putînd fi plictisitoare, chiar monotone. *Ele* sînt totuşi importante fiindcă *crează cadrul întregului siftware,* şi impun trăsăturile specifice modulelor ierarhic inferioare.

3. Pe parcursul elaborării acestui nivel am încercat să ne conformăm la *sugestia dată în Cap. 10, ca fiecare modul să fie cît se poate de autonom* evitîndu-se salturile de la unul la altul.

Astfel constatăm că procedurile **ERATICK, TOTSORT, TOTDAY** şi **SYNTH** au ramura de eroare identică, dar am înglobat-o în fiecare rutină, în ideea enunţată mai sus, precum şi animaţi de dorinţa de-a nu-i sustrage atenţia cititorului, care a avut oricum mult de răsfoit pentru a satisface referinţele făcute la specificaţia tehnică, şi la structurile de date.

4. Poate unii vor constata faptul că am folosit *etichete în dreptul instrucţiunilor* **RET,** lungind astfel rutinele cu cel puţin 2 octeţi prin includerea în corpul lor a unor salturi la adresa instrucţiunii **RET.**

Am ales şi această soluţie urmînd recomandările din Cap. 10, *ca fiecare subrutină să se termine într-un singur punct de ieşire* **RET,** şi acela să fie ultima instrucţiune din listă.

5. Înainte de-a încheia, sintetizăm lista rutinelor ierarhic inferioare, a căror specificaţie s-a făcut în §14.1.

a) **INPRICE** — §14.1.1. — 50%
b) **PRPRICE** — §14.1.1. — 50%
c) **NXPRICE** — §14.1.1. — 50%
d) **EMITBUYT** — §14.1.1. — 25%
e) **CLDISP** — §14.1.1. — 100%
f) **CNTFUNC** — §14.1.2. — 100%

g) **STORDISP** — §14.1.3. — 90%
h) **TRANSPAR** — §14.1.3. — 100%
i) **CLBUFDP** — §14.1.3. — 100%
j) **ERPRICE** — §14.1.4. — 50%
k) **EMITERAT** — §14.1.4. — 25%
l) **DISPNUM** — §14.1.2. — 90%
m) **INCODE** — §14.1.5. — 100%
n) **SORTADR** — §14.1.5. — 100%
o) **BCDCI** — §14.1.5. — 90%
p) **DISPSUM** — §14.1.5. — 90%
q) **PRTTICK** — §14.1.7. — 60%

14.2. Programe de prelucrare

Procedurile cuprinse în acest paragraf formează **nivelul ierarhic 1** al software-ului casei de marcat. Spre deosebire de nivelul ierarhic zero, care stabileşte cadrul (ambianţa) de lucru şi defineşte proceduri, *nivelul* **1** *va fi mai concret*, rezolvînd chiar dacă nu pînă la ultimul detaliu, funcţiile impuse prin specificaţia tehnică, folosind căile inaugurate în nivelul ierarhic zero.

14.2.1. Citirea unui preţ : INPRICE

Oricine se poate întreba de ce cele trei proceduri legate de pregătirea unui preţ : **citirea (INPRICE), prelucrarea (PRPRICE),** şi pregătirea următorului preţ **(NXPRICE)** nu formează corpul unei singure proceduri ? Răspunsul este simplu : procedura de emitere a unui bon de anulare **ERATICK** va folosi prima şi ultima procedură, dar miezul va diferi **(ERPRICE).**

A doua întrebare care se ridică se referă la rolul procedurii **NXPRICE.** De ce nu poate fi înglobată în **INPRICE** ? După analiza organigramelor capitolului precedent răspunsul se obţine şi în acest caz, relativ uşor.

■ În **INPRICE** se poate ajunge din două puncte de plecare : **repaosul interbon** şi **repaosul interpreţ.** Între cele două situaţii apare deosebirea esenţială că în repausul interbon **(MAINLOOP),** tastările **FUNC** (una sau două) care preced un eventual cod de marfă, vor fi deja prelucrate în momentul apelării procedurii **BUYTICK** (deci şi **INPRICE).** Plecînd din **repaosul interpreţ,** aceste operaţii ar trebui să fie efectuate de rutina **INPRICE.** Pentru a-i conferi o funcţie clar definită, simplificîndu-i astfel şi structura, am decis constituirea procedurii **NXPRICE.**

■ **NXPRICE** va funcţiona ca un moderator : graţie ei, procedura **INPRICE** va fi apelată totdeauna în momentul în care deja va fi fost tastat primul caracter din codul de sortiment, sau din valoarea preţului care urmează a fi introdus.

Plecînd de la aceste considerente rezultă descrierea de mai jos :

```
procedure inprice
1   begin
2       if nrfunc ≠ 0 then incode  {se citeşte de la tastatură codul şi încă o cifră sau punct}
3                     else implcode {codul este implicit 99}
4       repeat
5           begin
6               stordisp {cifra sau punctul se memorează şi se afişează}
7               repeat
8                   begin
9                       inkey
10                      if key = CLEAR then clbufdp {reiniţializare}
11                      if key = * then multop {pregătirea înmulţirii}
12                  end
13          until (key = cifră) V (key = punct) V (key = +)
14      end
15  until (key = +)
16  end
```

Programul aferent va mai preciza cîteva elemente.

```
1   INPRICE :   LD      D,A
2               LD      A,C
3               OR      A
4               LD      A,D
5               JR      Z,CODELESS
6   CODED :     CALL    INCODE
7   AFTERCOD :  CP      NRORPT
8               JR      C,PROCDIG
9               CALL    INKEY
10              JR      AFTERCOD
12  CODELESS :  LD      C,9
13              CALL    IMPLCODE
14              LD      HL,NRFUNC
15              LD      (HL),1
16  PROCDIG :   CALL    STORDISP
17  NEXTKEY :   CALL    INKEY
18              CP      CLEAR
19              CALL    Z,CLBUFDP
20              CP      ASTER
21              CALL    Z,MULTOP
22              CP      NRORPT
23              JR      C,PROCDIG
24              CP      PLUS
25              JR      NZ, NEXTKEY
26              RET
```

Adnotări :

Evenimentele importante legate de introducerea unui preț se regăsesc în programul de sus. Iată-le :

- Înainte de toate, *se verifică* (pe baza valorii lui **nrfunc** (aici în registrul C) *dacă caracterul* conținut în **A** face *parte din preț*, sau dintr-un cod de sortiment care precede prețul. Dacă prețul s-a tastat fără cod de marfă (C=0) atunci se va genera în mod implicit **codul 99** (conform specificației funcțiilor **F.1.12**). Secvența se regăsește în liniile *program* 12—13.
- *dacă* **nrfunc** *diferit de zero* înseamnă că prețul ce urmează a fi introdus, este precedat de cod de sortiment. *Intregul cod va fi citit de procedura* **INCODE** (linia 6).
- În afara misiunii de a citi cel de-al doilea caracter din cod, și eventual să permită folosirea tastei **CLEAR** pentru corectarea erorilor din cod, **INCODE** va trebui să genereze și un indicator, preferabil în memoria **RAM**, care va informa programele de prelucrare **PRPRICE** și **ERPRICE** asupra numărului de tastări **FUNC** care a precedat codul respectiv de sortiment (reamintim că o tastare **FUNC** (**nrfunc**=1) înseamnă vînzare, deci bani intrați în casă, iar 2 tastări **FUNC** (**nrfunc**=2) reprezintă răscumpărare de ambalaj, deci bani ieșiți din casă). **INCODE** *va genera în* **RAM** *un indicator pe care-l vom numi* **NRFUNC.** Această variabilă *va conține numărul de tastări* **FUNC,** *care au precedat introducerea unui cod de sortiment.* Și în cazul în care prețul s-a tastat fără cod de sortiment, **NRFUNC** trebuie actualizat, de astă dată de către **INPRICE** (vezi liniile *program* 14—15)
- *La adresa* **PROCDIG** (linia 16) se ajunge mereu cu *caractere valide* (cifră sau punct) *ale prețului.* Ele se depun în **KEYBUF** și se afișează **(LEDBUF).**
- Secvența *program* cuprinsă între liniile 17—26 citește următoarele caractere ale prețului, așteptînd sosirea unui terminator :
  - **dacă se tastează CLEAR,** se șterge afișajul și bufferul de intrare **KEYBUF,** după care se va relua introducerea prețului, (deja fără cod) linia 19, **CLBUFDP ;**
  - **dacă se tastează** semnul înmulțirii "*", se vor lua măsuri pregătitoare prin procedura **MULTOP** (linia 21). **MULTOP** va trebui să preia prețul, să efectueze asupra lui toate verificările și să-l convertească în format **BCD** și să-l depună într-unul din regiștri **BCD (TMPBCD** sau **DATBCD).** Totodată, **MULTOP** va înscrie un indicator, indicator care va fi verificat la apariția unei taste "+". Dacă indicatorul de înmulțire va fi activ atunci, după tastarea semnului "+" va trebui efectuată o înmulțire.
- Ieșirea din rutina **INPRICE** se va face doar la apariția tastei "+" vezi liniile 24—26 din *program.*

14.2.2. Prelucrarea unui preț : PRPRICE

La ieșirea din rutina **INPRICE,** în **KEYBUF** se află un număr cu max. 8 cifre semnificative, care ar trebui să reprezinte un preț de produs.

■ Înainte de a aduna valoarea lui la totalul clientului, și de a-l imprima, va trebui să se verifice corectitudinea sa sintactică, pentru ca apoi să fie normalizat (2 cifre zecimale), iar doar după aceea convertit în format BCD, pentru a putea fi manipulat (adunat, scăzut, înmulțit). Abia după ce s-au efectuat aceste manevre pregătitoare, se vor putea întreprinde activitățile specifice de casă de marcat.

Iată algoritmul propus :

```
procedure prprice
1   begin
2   syntan {se analizează prețul introdus}
3          if "corect" then
4             begin
5             prelproc {prețul se transformă în BCD și dacă este indicat, se execută înmulțirea}
```

```
6          addsort  {se adună la totalul de sortiment
                    corespunzător codului}
7          opforbuy {se execută operațiile legate de bonul clien-
                    tului : prețul se adună la totalul cli ntu-
                    lui, sau se scade din totalul clientului. noul
                    total se afișează, iar prețul se imprimă ; e bon}
8                   end
9          end
```

Rutina aferentă va fi :

```
1  PRPRICE :   CALL     SYNTAN
2              JR       NZ,ENDPRPR
3              CALL     PRELPROC
4              CALL     ADDSORT
5              LD       HL,PRTBUF+15
6              LD       (HL),PLUS
7              CALL     OPFORBUY
8  ENDPRPR :   RET
```

Adnotări :

- Procedura **SYNTAN** *va verifica* corectitudinea șirului de cifre și puncte, din **KEYBUF**, și dacă îl, va găsi corect, îl *va depune* în **EBCD** (vezi §13.3.1). În *caz de eroare*, ea va returna rutinei apelante flagul **Z** resetat (Z=0). În acest caz, **PRPRICE** *va fi ocolit complet.* (Ca eroare posibilă amintim un număr cu mai multe puncte zecimale).
- Procedura **PRELPROC** *va converti* numărul din format **BCD extins** în format **BCD**, și-l *va depune* în **DATBCD.** *Dacă* indicatorul de înmulțire va fi activ, *va efectua înmulțirea* celor două numere **(DATBCD** și **TMPBCD).**
- Procedura **ADDSORT** adună la valoarea curentă a registrului **BCD** afectat sortimentului, care are codul specificat în **CODE,** numărul **DATBCD.**
- Procedura **OPFORBUY** *va completa totalul clientului* (registrul **GUESTBCD)** și *va imprima un rînd de preț,* marcînd semnul **"+",** în dreptul cifrei celei mai puțin semnificative (liniile 5—7) ; se afișează suma curentă a clientului, pe dispozitivul de afișaj. Ea va fi aceea, care *va* folosi variabila **RAM NRFUNC,** specificată la prezentarea rutinei **INPRICE** pentru a *decide sensul de manevrare a banilor intrați în casă sau ieșiți din casă.*

14.2.3. Anularea unui preț : ERPRICE

Procedura **ERPRICE** apelată în procedura de emitere a unui bon de anulare **(ERATICK)** va avea aceeași structură ca și **PRPRICE,** diferențele constă în :

- *Linia de preț* imprimată va fi *marcată* cu **"A"** în loc de **"+"** .
- *Suma* introdusă *se va scade* din totalul de sortiment **(SUBSORT).**
- *Va* trebui să fie prevăzută cu un test suplimentar care *interzice operația de anulare, dacă suma* totală aferentă *oricărui sortiment* implicat *devine negativă.*

Listingul comentat al acestei proceduri se regăsește în Cap. 17, pag. 1—23.

14.2.4. Pregătirea următorului preț : NXPRICE

Funcțiile acestei proceduri au fost specificate la descrierea rutinei **INPRICE** (§14.2.1) vom trece direct la elaborarea algoritmului aferent.

```
   procedure nxprice
1    begin
2        nrfunc : = 0
3        maxfunc : = 2
4        repeat
5            begin
4                inkey
7                if key = FUNC then cntfunc   {contorizare FUNC
                                              {și afișare
8            end
9        until (key=cifră) ∨ (key=punct) ∨ (key=TOTAL)
10       clbufdp
11   end
```

Rutina în limbaj de asamblare rezultă :

```
 1  NXPRICE :   LD      C,0
 2              LD      B,MAXFUNC
 3  BEGPR :     CALL    INKEY
 4              CP      FUNC
 5              CALL    Z,CNTFUNC
 6              CP      NRORPT
 7              JR      C,ENDNXPR
 8              CP      TOTAL
 9              JR      NZ,BEGPR
10  ENDNXPR :   CALL    CLBUFDP
11              RET
```

Adnotări :

- În secvența cuprinsă între liniile 3—9 identificăm *repaosul interpreț.*
- Din această secvență nu se va putea *ieși* decît *prin* apăsarea unei *cifre* sau *punct* (liniile 6, 7, 10, 11), considerat a fi primul caracter al unui cod de produs sau al unui preț, *sau* apăsînd tasta **TOTAL**, care va provoca imprimarea efectivă a bonului client normal, sau de anulare (liniile 8—11).
- **NXPRICE** *va limita tastările succesive* **FUNC**, afișînd o numărare cu secvența : 1, 2, 1, 2,... impusă prin valoarea maximă admisă de tastări succesive : **MAXFUNC**=2. (Reamintim că din repaosul interpreț (conform specificației funcționale) nu se vor putea lansa funcții speciale ale casei de marcat (cele care ar necesita mai mult decît 2 tastări succesive **FUNC**).

14.2.5. Citirea codului de sortiment : INCODE

■ Procedura de citire a unui cod de sortiment, format obligatoriu din 2 cifre, **INCODE**, a fost specificată funcțional la prezentarea procedurii de citire a unui preț **INPRICE** (vezi §14.2.1)

Structura ei fiind foarte simplă, ne permitem să elaborăm direct programul în limbaj de asamblare :

```
 1  INCODE :     LD      HL,NRFUNC
 2               LD      (HL),C
 3               LD      B,SCLEN
 4  CODDIG :     CP      POINT
 5               JR      C,PRCODDIG
 6               CALL    INKEY
 7               JR      CODDIG
 8  PRCODDIG :   CALL    STORDISP
 9               CALL    INKEY
10               CP      CLEAR
11               CALL    Z,CLEARCOD
12               DJNZ    CODDIG
13               LD      HL,KEYBUF+BUFLEN-SCLEN
14               LD      DE,CODE
15               LD      BC,SCLEN
16               LDIR
17               CALL    CLBUFDP
18               RET
19  CLEARCOD :   CALL    CLBUFDP
20               LD      B,SCLEN+1
21               RET
```

Adnotări :

- Foarte importantă acțiunea din liniile 1—2 : **contorul din RAM** al tastărilor succesive **FUNC**, *este actualizat cu* valoarea cuprinsă în **C**.
- **SCLEN** (Sort Code **LEN**gth) precizează *lungimea codului de sortiment* (2 în cazul nostru)
- În *prima buclă* (liniile 4—7) se *filtrează toate tastele care diferă de cifre* ; știind că prima tastă după **FUNC**, va trebui să fie obligatoriu cifră, *(nu se va admite nici* **CLEAR**).
- În **secvența principală** (liniile 8—11) codul cifrei tastate se vizualizează și se depune în **KEYBUF** (STORDISP).
- În cazul în care se tastează **CLEAR**, se *apelează rutina* **CLEARCOD** : care *va* șterge cele 2 buffere **KEYBUF** și **LEDBUF**, și va reinițializa contorul de evenimente din **B**.
- Citirea cifrelor de cod continuă pînă cînd **B**=0 (linia 12).
- **INCODE** se termină printr-o secvență care *transpune codul* tastat, *din* **KEYBUF**, *în* locul de destinație **CODE** (din **EDBUF**).
- **BUFLEN** este lungimea bufferului **KEYBUF**.

14.2.6. Generarea implicită a codului de sortiment : IMPLCODE

Misiunea acestei proceduri este deosebit de simplă. Ea va trebui să înscrie în zona dedicată codului de sortiment (**CODE** din **EDBUF**) **codul 99**, rezervat pentru mărfurile a căror preț se introduce fără specificarea codului de sortiment.

Listingul comentat al rutinei **IMPLCODE** se găsește în Cap. 17. pag. 1—26.

Această procedură va fi apelată la detectarea apăsării tastei **FUNC**

■ Ea va incrementa numărătorul **nrfunc** (din registrul **C**), și îi va afișa valoarea în extremitatea din stînga a dispozitivului de afișaj.

■ Dacă registrul **C** depășește o valoarea impusă la apelare (**funcnum** sau **maxfunc** în **B**), atunci **CNTFUNC** va reinițializa contorul : C = 1.

```
    procedure cntfunc
1       begin
2           nrfunc : = nrfunc+1
3           if nrfunc < funcnum then nrfunc = 1
4           cldisp
5           begin
6           „citește codul de comandă segmente al nfrunc din LEDGEN"
7           „depune codul obținut în LEDBUF+0"
8           end
9           sonerie
10      end
```

Implementarea acestei proceduri nu ridică probleme. O redăm totuși, pentru exemplificarea accesului la generatorul de coduri pentru afișaj **LEDGEN**.

```
1  CNTFUNC :   PUSH    AF
2              INC     C
3              LD      A,B
4              CP      C
5              JR      NC,DISPNR
6              LD      C,1
7  DISPNR :    CALL    CLDISP
8              LD      B,0
9              LD      HL,LEDGEN
10             ADD     HL,BC
11             LD      B,A
12             LD      A,(HL)
13             LD      DE,LEDBUF+0
14             LD      (DE),A
15             LD      HL,FCSDFRE
16             LD      A,FCSDTIME
17             CALL    BEEP
18             POP     AF
19             RET
```

Adnotări :

- *Incrementarea* **nrfunc** se face în linia 2.
- *Testarea* valorii *limitei impuse*, și o eventuală *reinițializare* a numărătorului **nrfunc** se fac în liniile 3—6.
- În secvența 8—10 se *calculează adresa din generatorul de coduri comandă segmente*, la care se află locat codul cifrei egale cu **nrfunc.**
- *Codul de comandă* se depune *în* **LEDBUF** pe poziția *extremă stîngă*, în liniile 13—14.
- Ultima secvență (15—17) *emite un sunet scurt* pentru a marca fiecare tastare **FUNC.**

14.2.8. Operații pentru client : OPFORBUY

■ Procedura **OPFORBUY** se apelează în momentul în care ultimul preț introdus se află, validat, în **DATBCD**, iar **NRFUNC** indică tipul operației de efectuat. Ea va efectua *strict operații pentru client* (totalul de sortiment a fost deja actualizat în PRPRICE sau ERPRICE, prin apelul uneia din rutinele ADDSORT sau SUBSORT).

```
    procedure opforbuy
1       begin
2           if nrfunc = 1
3               then if code > 9
4                   then „se adună prețul introdus la totalul clientului"
5               else if code ≤ 9
6                   then
7                       begin
8                           „se schimbă în bufferul de tipărire + cu
                            —"
9                           „se scade prețul introdus din totalul clien-
                            tului"
10                      end
11              if „nu s-a detectat eroare"
12                  then
13                      begin
14                          „se afișează noul total al clientului"
15                          „se tipărește prețul introdus pe bon"
16                      end
17                  else
18                      begin
19                          „se scade prețul introdus din totalul de
                            sortiment corespunzător"
20
21                          „se afișează vechiul total al clientului"
22                      end
23      end
```

Adnotări :

- *Dacă* **NRFUNC** = 1 *atunci codul de sortiment* va trebui să fie cuprins în gama de valori *10—99. In caz contrar*, operatorul a greșit, caz în care operațiile deja efectuate trebuie anulate și *comanda va fi rejectată* (liniile 18—22 din descriere).
- *Dacă* **NRFUNC** = 2, *atunci codul de sortiment* va trebui să fie, cuprins între *0—9. In caz contrar* se va semnala aceeași *eroare operator.*
- **Esența procedurii** este cuprinsă în liniile 4, 9, 14, 15 ale *descrierii.*
- **Adunarea** și **scăderea** vor apela rutinele de aritmetică **BCD.**
- **Vizualizarea** se va face cu rutina **DISPSUM.**
- **Imprimarea** liniei de preț se va face cu **PRTLINE.**

Listingul comentat al rutinei **OPFORBUY** se găsește în Cap. 17 pag. 1—27, 1—28.

14.2.9. Emiterea bonului client normal : EMITBUYTICK

■ Această procedură va fi declanşată de apăsarea tastei **TOTAL** în *repaosul interbon* **(MAINLOOP)** sau în *repaosul interpreţ* **(NXPRICE).**

■ Apelul ei se face din procedura BUYTICK.

Activităţile pe care le întreprinde sînt redate în descrierea formală care urmează.

```
1 begin
2         „se incrementează numărătorul bonurilor"
3         „se adună totalul clientului la totalul zilei"
4         „totalul clientului se afişează"
5         „se imprimă bonul clientului"
6         „totalul clientului se reiniţializează cu 0"
7 end
```

Notăm faptul că imprimarea va fi efectuată cu rutina **PRTTICK :** după listarea mesajului **"* VA MULŢUMIM" *,** se va imprima un rînd gol, urmat de antetul bonului următor **(UNIT.NR., CASA NR.),** secvenţă care rezultă din structura tabelei **EDBUF** (vezi fig. 13.10)

Listingul comentat al procedurii se regăseşte în Cap. 17. pag. 1—31.

14.2.10. Emiterea bonului client normal : EMITERATICK

■ Cu toate diferenţele de esenţă pe care operaţia de emitere a unui bon de anulare le comportă, structura acestei proceduri va fi identică cu cea a procedurii **EMITBUYTICK.**

```
procedure emiterat
1     begin
2         „se incrementează numărătorul bonurilor"
3         „se scade totalul clientului din totalul zilei"
4         „totalul clientului se afişează"
5         „se imprimă bonul de anulare"
6         „totalul clientului se reiniţializează cu 0"
7     end
```

Listingul comentat al acestei rutine se găseşte în Cap. 17, pag. 1—32.

14.2.11. Imprimare din EDBUF : PRTTICK

Procedura **PRTTICK** specificată funcţional în §14.1.5. şi §14.1.7. va trebui să imprime conţinutul anumitor linii din **EDBUF.**

■ Liniile de imprimat se vor selecta pe baza unei măşti de selecţie (vezi §13.2.4.).

■ Imprimarea efectivă se va face prin **PRTLINE,** care va imprima pe cele două fişii ale casei de marcat, mesajul care fusese în prealabil transpus din **EDBUF,** prin procedura **TRANLINE.**

■ Un caz aparte îl constituie **PRTTICK**, lansat din **TOTSORT**, cînd linia a 6-a din EDBUF ($ID_0 = 86_H$) va trebui să fie imprimată de 100 de ori, după ce în prealabil zona afectată codului de sortiment fusese încărcată cu numărul de cod crescător, iar în zona de preţ (vezi fig. 13.10) s-a transferat totalul aferent. Această iteraţie va fi rezolvată în procedura SYNTTOTS.

```
procedure prttick
1   begin
2       contor : = numărul liniilor din tabela EDBUF
3       repeat
4           begin
5               if „linia curentă face parte din bonul în curs de tipărire"
6                   then if (este linia totalului) ∧ (bon de tip sin-
                    tează)
7                       then
8                           synttots {se imprimă cele 100 de / totaluri de sortiment}
9                       else
10                          tranline
11                          prtline {se imprimă linia curentă}
12                          "se trece la linia următoare"
13                          contor : = contor — 1
14          end
15      until contor = 0
16  end
```

```
1   PRTTICK :   LD      B,NREDLNS
2               LD      IX,EDBUF+2
3               LD      DE,EDLLEN
4   EDLINE :    LD      A,(IX—1)
5               AND     C
6               JR      Z,NEXTEDLN
7               LD      A,C
8               CP      SYNTMASK
9               JR      NZ,NORMLINE
10              LD      A,(IX—2)
11              CP      86H
12              JR      NZ,NORMLINE
13              CALL    SYNTTOTS
14              JR      NEXTEDLN
15  NORMLINE :  CALL    TRANLINE
16              CALL    PRTLINE
17  NEXTEDLN :  ADD     IX,DE
18              DJNZ    EDLINE
19              LD      HL,EDBFRE
20              LD      A,EDBTIME
21              CALL    BEEP
22              RET
```

Adnotări :

- **Decizia de imprimare** sau **neimprimare** a unei linii din **EDBUF**, formulată în *descriere* în liniile 6, 7, 9 se regăsește în *program* între liniile 7—12.
- *Simbolurile utilizate* au următoarea semnificație :
  NREDLNS — (NumbeR of **ED** LiNeS) — numărul total al liniilor din **EDBUF** — în registrul **B** ($0E_H$).
  EDLLEN — (**EDL**ine **LEN**ght) — lungimea unei linii în **EDBUF** — în registrul **DE** (0011_H)
  SYNTMASK — *masca de selecție* pentru liniile care vor fi imprimate în procedura **SYNTHi**
- Registrul index **IX** *va indica* mereu *începutul util* (imprimabil) *al unei linii* din **EDBUF** (vezi liniile program 2 și 17).
- **IX — 1** indică ID_1, iar **IX — 2** pontează pe ID_0 (vezi fig. 13.10).
- Procedura **TRANLINE** *transferă* linia curentă (partea utilă) *din* **EDBUF** *în* **PRTBUF** după cum urmează :

```
TRANLINE :      PUSH    BC
                PUSH    DE
                PUSH    IX
                POP     HL
                LD      DE,PRTBUF+1
                LD      BC,EDLLEN—2
                LDIR
                POP     DE
                POP     BC
                RET
```

- Dacă imprimarea liniilor din **EDBUF** a ajuns la linia marcată prin $ID_0 = 86_H$, și masca de selecție indică **SYNTH**, atunci se va declanșa procedura **SYNTTOTS** (liniile program 13—14).

```
   procedure synttots
1    begin
2        code :=0
3        cnt :=100
4        repeat
5            begin
6                ,,totalul de sortiment corespunzător codului curent se
                 transferă în bufferul de tipărire".
7                ,,se imprimă"
8                code/: = code + 1
9                cnt : = cnt—1
10           end
11       until cnt = 0
12   end
```

```
1   SYNTTOTS :      PUSH    DE
2                   PUSH    BC
3                   LD      C,0
4                   CALL    IMPLCODE
5                   LD      B,100
6   NSORTT :        PUSH    BC
7                   CALL    SORTADR
8                   LD      DE,PRTBUF+4
```

```
 9          CALL    BCDCI
10          LD      HL,CODE
11          LD      DE,PRTBUF+1
12          LD      BC,SCLEN
13          LDIR
14          LD      A,ASTER
15          LD      (PRTBUF+15),A
16          CALL    PRTLINE
17          LD      B,SCLEN
18          LD      HL,CODE+SCLEN-1
19          CALL    EBCDINC
20          POP     BC
21          DJNZ    NSORTT
22          POP     BC
23          POP     DE
24          RET
```

Adnotări :

- *Inițializarea* celor două variabile **code** și **cnt** din liniile 2, 3 ale *descrierii*, se regăsesc în liniile 3—5 ale *programului*.
- **Suma (BCD)** corespunzătoare *sortimentului curent*, avînd codul **code**, se transformă în cod intern și se *depune în* **PRTBUF** (liniile *program* 7, 8, 9)
- Valoarea curentă a codului de sortiment **code**, se transferă în **PRTBUF**, la începutul liniei (liniile program 10—13).
- *Linia imprimată* se *marchează cu* "x" în dreptul cifrei celei mai puțin semnificative.
- *Valoarea codului de sortiment* fiind exprimată în cod intern, *se incrementează* prin secvența program cuprinsă între liniile 17—19.
- Procedura **EBCDINC** *incrementează* un număr scris în format **EBCD** indicat prin **HL**, avînd lungimea specificată în **B**.
- Acțiunea **buclei principale** (liniile 6—21) se reia pînă cînd **B=0** (100 de ori).

Listingul comentat al rutinelor din acest paragraf se regăsește în Cap. 17, p. 1—33 —1—35.

4.2.12. Localizarea registrului de sortiment : SORTADR

■ Procedura **SORTADR** localizează un registru în tabela **SORTTOT** pe baza valorii din **CODE.**

Redăm descrierea formală a procedurii :

```
procedure sortadr
    begin
        „numărul ebcd din zona code se transformă în binar"
        „se localizează registrul căutat, aplicînd relația 13.2., adresa de început
        în hl"
    end
```

În Cap. 17, pag. 1—29 se regăsește listingul comentat al procedurii.

14.2.13. Actualizarea totalului de sortiment (+/—) : ADDSORT/SUBSORT

■ Cele două proceduri adună (respectiv scad) prețul reprezentat în format **BCD**, aflat în **DATBCD**, la suma curentă din **SORTTOT**, pe baza codului de sortiment solicitat, din **CODE**.

Iată structura posibilă a acestor rutine :

```
ADDSORT :   CALL    SORTADR ; (sau SUBSORT)
            PUSH    HL
            CALL    MOVTMP
            CALL    ADDBCD ; (sau SUBBCD)
            POP     DE
            CALL    TMPMOV
            RET
```

● Rutinele **MOVTMP** respectiv **TMPMOV** mută o valoare BCD (de 5 byte) "în" și "din" **TMPBCD**. **ADDBCD** și **SUBBCD** efectuează operațiile aritmetice "+" sau "—" asupra regiștrilor **BCD**, **TMPBCD** și **DATBCD**.

Aici *se termină* prezentarea **nivelului ierarhic 1 al software**-ului casei de marcat. Mai mult sau mai puțin, fiecare din rutinele prezentate au fost particulare proiectului considerat. Odată cu acest paragraf *se termină și referințele făcute la specificația tehnică. Modulele* care se vor prezenta *în paragrafele următoare* sînt fie cu *caracter general* cum ar fi **aritmetică BCD, conversii de coduri,** *sau* reprezintă *detalii* (intimități) *de programare.*

14.3. Aritmetica BCD cu virgulă fixă

Așa cum am prezentat în Cap. 13, la elaborarea structurilor de date, ne propunem să scriem un pachet de **programe aritmetice,** care să trateze **numere BCD compacte** (2 cifre/octet), avînd **lungimea de 5 octeți,** dintre care cel de-al 5-lea octet este partea zecimală. Numerele reprezentate vor fi fără *semn.*

Rutinele de *adunare, scădere* și *înmulțire* vor folosi regiștri **TMPBCD** și **DATBCD**, *rezultatul* generîndu-se în **TMPBCD**.

14.3.1. Adunarea a două numere : ADDBCD

Acțiunea procedurii este :

$$\mathbf{TMPBCD} := \mathbf{TMPBCD} + \mathbf{DATBCD} \qquad (14.2.)$$

■ Locarea octeților în regiștri **BCD** respectă structura :
REGBCD + 0 : octetul cel mai semnificativ
REGBCD + 4 : octetul cel mai puțin semnificativ
■ În cadrul unui octet, $D_7 — D_4$ conține cifra cea mai semnificativă.

Dacă **HL** pontează pe octetul cel mai puţin semnificativ din **DATBCD**, iar **DE** pe acelaşi octet în **TMPBCD** şi **B** conţine lungimea numărului BCD (exprimat în octeţi), atunci secvenţa care urmează va efectua *adunarea* celor două numere, generînd rezultatul în **TMPBCD**.

```
1                XOR     A
2   ADDBYTE :    LD      A,(DE)
3                ADC     A,(HL)
4                DAA
5                LD      (DE),A
6                DEC     DE
7                DEC     HL
8                DJNZ    ADDBYTE
```

Pentru ilustrarea acţiunii acestei secvenţe vom elabora o *schemă sinoptică* (fig. 14.9.), pe care vom parcurge de 2 ori bucla program cuprinsă între liniile 2—8, ilustrînd evoluţia regiştrilor BCD în cazul adunării numerelor **1484,25** şi **347,50**.

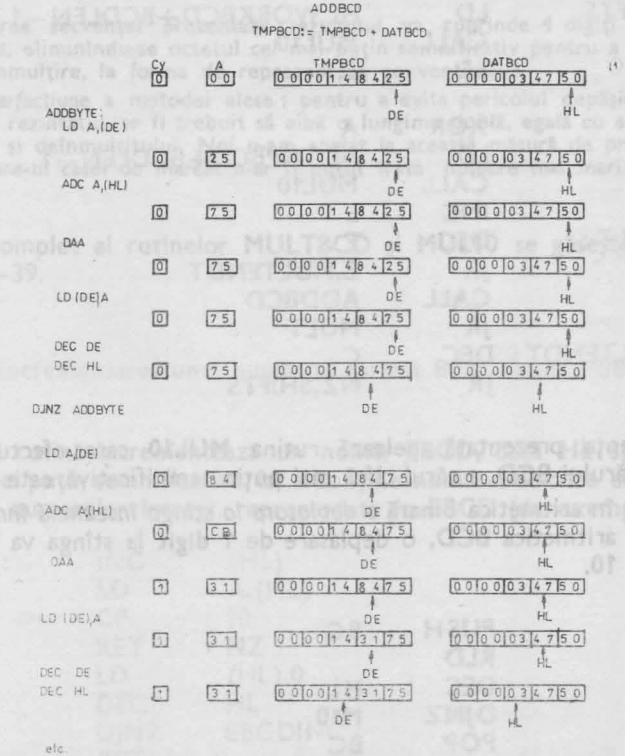


Fig. 14.9. Schema sinoptică de acţiune a rutinei ADDBCD

14.3.2. Scăderea a două numere : SUBBCD

■ Acțiunea acestei rutine va decurge în aceleași condiții ca și cea a rutinei **ADDBCD**, diferența constînd în operația aritmetică de bază executată : **SBC A,(HL)** în loc de **ADC A,(HL)**.

■ În cazul unui rezultat negativ : $C_Y = 1$

În Cap. 17, pag. 1—37 se regăsește listingul comentat al acestei rutine.

14.3.3. Înmulțirea a două numere : MULTBCD

■ Presupunem că în momentul apelului, **TMPBCD** conține **înmulțitorul**, iar **DATBCD deînmulțitul**. Pentru a genera **rezultatul** în **TMPBCD**, vom disloca înmulțitorul din **TMPBCD** într-un registru de lucru nou creat : să-l numim **WORKBCD**. După ștergerea registrului **TMPBCD** inițializăm numărătorii :

B — lungimea unui număr **BCD** (5)

C — numărul de cifre al numărului **BCD** ($2 \times 5 = 10$)

Secvența redată în continuare va efectua înmulțirea numerelor :

```
1     SHIFTS :    LD      HL,WORKBCD+BCDLEN-1
2                 CALL    MUL10
3                 LD      E,A
4
5                 XOR     A
6                 LD      HL,TMPBCD+BCDLEN-1
7                 CALL    MUL10
8                 INC     E
9     MULT :      DEC     E
10                JR      Z,MULTENDT
11                CALL    ADDBCD
12                JR      MULT
13    MULTENDT :  DEC     C
14                JR      NZ,SHIFTS
```

■ Secvența prezentată apelează rutina **MUL10** care efectuează înmulțirea cu **10** a numărului **BCD**, a cărui cifră mai puțin semnificativă este pontată de **HL**.

Așa cum în aritmetica **binară** *o deplasare la stînga înseamnă înmulțirea* cu baza **2**, tot așa în aritmetica **BCD**, o deplasare de **1 digit** la stînga va însemna înmulțire cu baza **10**.

```
MUL10 :     PUSH    BC
M10 :       RLD
            DEC     HL
            DJNZ    M10
            POP     BC
            RET
```

Acesta este un exemplu clasic de utilizare a puternicei instrucțiuni **Z80, RLD.**

Acțiunea buclei din **M10** o ilustrăm în fig. 14.10.

Revenind la rutina **MULTBCD,** constatăm că înmulțirea deînmulțitului cu fiecare cifră a înmulțitorului se face prin adunări succesive (liniile 9—12).

Inmulțirea nu se efectuează dacă digitul curent al înmulțitorului, care pe urma deplasării lui **WORKBCD** (liniile 1—2) apare în **A,** este **0.**

La fiecare parcurgere a buclei principale, conținutul registrului rezultat **TMPBCD,** este deplasat cu un digit la stînga.

Pentru ilustrarea acțiunii secvenței de înmulțire am realizat *schema sinoptică* din fig. 14.11.

Fig. 14.10. Schema sinoptică de acțiune a rutinei MUL10

Adnotări :

- După efectuarea secvenței prezentate *rezultatul* va cuprinde 4 digiți zecimali. El *va trebui trunchiat,* eliminîndu-se octetul cel mai puțin semnificativ pentru a aduce numărul rezultat din înmulțire, la forma de reprezentare convențională.
- Rezultă *o inperfecțiune* a metodei alese : pentru a evita pericolul depășirii, registrul în care se crează rezultatul, ar fi trebuit să aibă o lungime dublă, egală cu suma lungimilor înmulțitorului și deînmulțitului. Noi n-am apelat la această măsură de protecție, fiindcă oricum software-ul casei de marcat n-ar fi putut trata numere mai mari decît formatul impus.

Listingul complet al rutinelor **MULTBCD** și **MUL10** se găsește în Cap. 17, pag. 1—38, 1—39.

14.3.4. Incrementarea unui număr în format BCD extins : EBCDINC

■ Această rutină incrementează un număr **EBCD,** dacă **HL** specifică octetul (cifra) cea mai puțin semnificativă, iar **B** conține numărul de cifre ale numărului. Ea se va aplica numerelor întregi, reprezentate în **EBCD** (de exemplu : **CODE).**

```
1  EBCDINC :   INC     (HL)
2              LD      A,(HL)
3              CP      10
4              RET     NZ
5              LD      (HL),0
6              DEC     HL
7              DJNZ    EBCDINC
8              RET
```

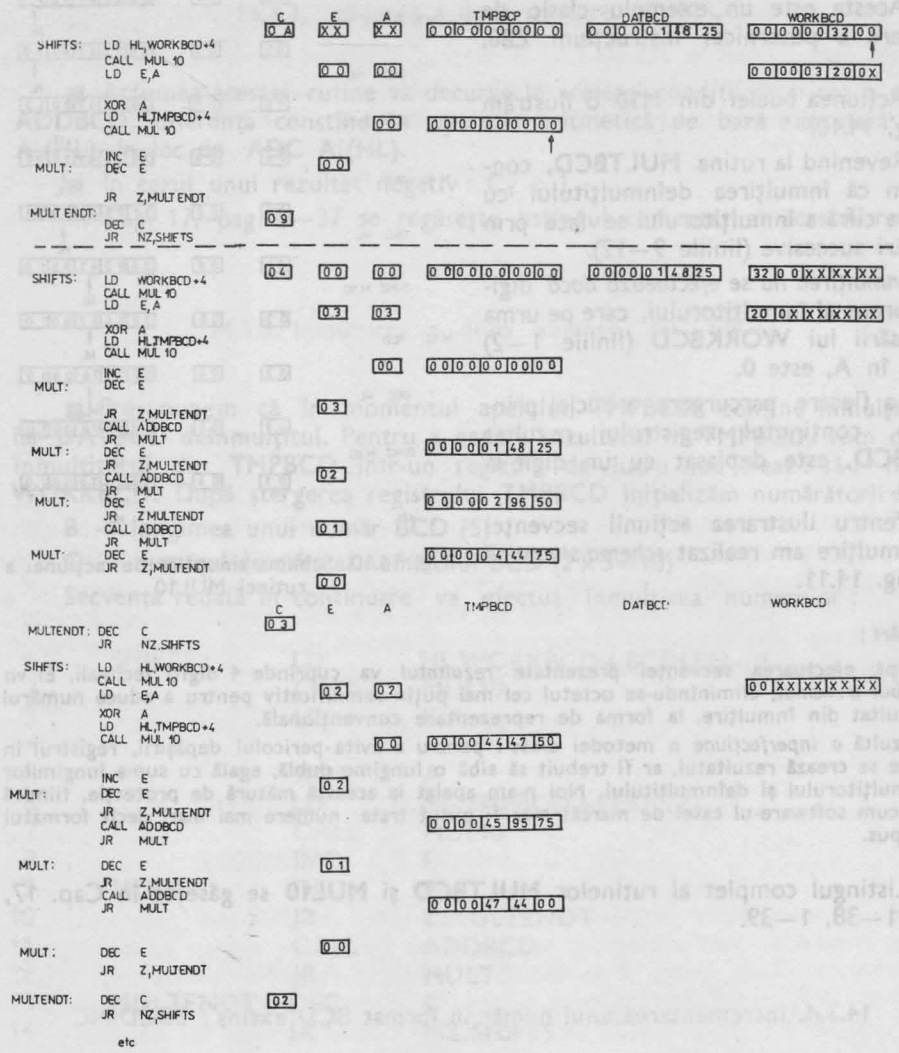


Fig. 14.11. Schema sinoptică de acțiune a rutinei MULTBCD

Dacă cifra mai puțin semnificativă nu depășește valoarea 9 după incrementare, atunci rutina se termină în linia 4. În caz contrar se va trata următoarea cifră semnificativă a numărului.

În prezentul paragraf nu ne-am propus elaborarea unui set complet de programe aritmetice BCD. Le-am tratat pe acelea, care în casa de marcat sînt absolut necesare. Sperăm ca ele să fie reușit să prezinte totuși cîteva noțiuni, care pot fi utile.

14.4. Analiză sintactică și conversii de coduri

Pentru ca prețurile introduse de la tastatura casei de marcat să poată fi prelucrate și apoi afișate, respectiv imprimate, sînt necesare o serie de conversii.

Prima operație este aceea de a valida sintactic numărul introdus de la tastatură.

14.4.1. Analiza și conversia numerelor introduse de la tastatură: SYNTAN

■ Numerele tastate nu vor putea conține mai mult decît un punct zecimal. Numărul introdus trebuie normalizat: partea zecimală va avea obligatoriu 2 cifre, indiferent că ele au fost tastate sau nu.

■ Procedura **SYNTAN** va efectua aceste operații asupra numărului primit în **KEYBUF**, depunînd rezultatul, (din care se filtrează și punctul zecimal) în zona **EBCD**. (Vezi definiția structurii în §13.3.1. fig. 13.12.).

Pentru exemplificarea modului în care **SYNTAN** normalizează numerele din **KEYBUF** am recurs la o *schiță* (vezi fig. 14.12). Se poate remarca tratarea distinctă a părții întregi și a celei zecimale.

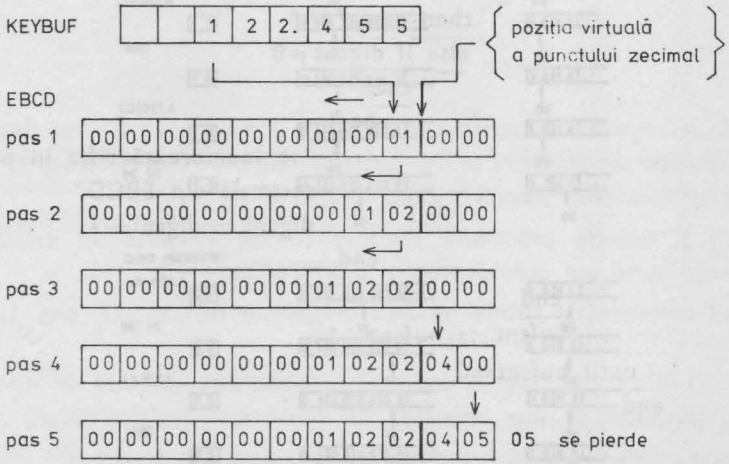


Fig. 14.12. Schema sinoptică de acțiune a rutinei SYNTAN

Prezentăm descrierea în limbaj de nivel înalt a procedurii **SYNTAN.**

```
procedure syntan
    begin
        „se inițializează EBCD cu "0"
        bufcnt : = lungimea bufferului KEYBUF
        errorf : = 0
        „se caută primul octet din KEYBUF diferit de spațiu"
        if este cifră
          then
          begin
              „se memorează cifra în partea întreagă din EBCD"
              while (bufcnt≠0) ∧ „octetul curent din KEYBUF nu conține do
                                  punct"
              begin
                  „se ia octetul următor din KEYBUF"
                  bufcnt : = bufcnt—1
                  "se memorează cifra în partea întreagă din EBCD"
              end
          end
        if bufcnt≠0
        then
          begin
            digcnt :=2
            repeat
                begin
                    „se ia octetul următor din KEYBUF"
                    if conține punct
                      then errorf :=1
                      else if digcnt≠0
                          then
                            begin
                                "se memorează cifra în partea
                                zecimală din EBCD"
                                digcnt :=digcnt—1
                            end
                end
                bufcnt :=bufcnt—1
            until bufcnt=0
          end
    end
```

În Cap. 17, pag. 1—42, 1—43 se găsește listingul comentat al procedurii **SYNTAN.**

14.4.2. Impachetarea numerelor EBCD în format BCD : EBCDBCD

■ *Misiunea* acestei rutine *este cea de a împacheta un număr* reprezentat în format **BCD extins** (o cifră/octet), situat în **EBCD**, în format **BCD compact** (2 cifre/octet) și al depune în registrul **DATBCD.**

■ Vom folosi 2 indicatori :

HL — pontează pe adresa de început a bufferului sursă : **EBCD**

DE — pontează pe adresa de început a bufferului destinație : **DATBCD**

■ *Contorizarea* evenimentelor se va face *în* registrul **B** pe care-l *inițializăm la* lungimea unui număr **BCD compact** *(5 byte).*

În aceste condiții secvența ce urmează, va efectua conversia propusă :

```
1     PACKBYTE :   LD      A,(HL)
2                  INC     HL
3                  RLCA
4                  RLCA
5                  RLCA
6                  RLCA
7                  RRD
8                  LD      (DE),A
9                  INC     HL
10                 INC     DE
11                 DJNZ    PACKBYTE
12                 RET
```

În centrul atenției se află instrucțiunea **RRD** (Rotate Right Digit), prezența căreia simplifică considerabil problema.

În fig. 14.13 prezentăm dinamica secvenței **PACKBYTE** din **EBCDBCD.**

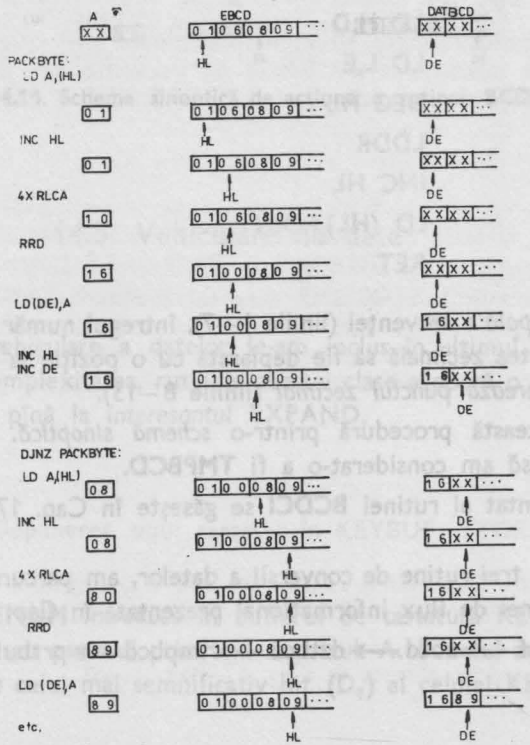


Fig. 14.13. Schema sinoptică de acțiune a rutinei PACKBYTE

14.4.3. Conversia inversă a numerelor : BCDCI

■ Această rutină *despachetează* numărul de la adresa indicată de **HL** *în cod intern, şi îl depune* într-o zonă începînd cu adresa **DE.**

■ Ea generează şi punctul zecimal, substituind zerourile nesemnificative cu spaţii.

Implementarea ultimei cerinţe nefiind o problemă, ne vom concentra atenţia asupra secvenţei de conversie **BCD** în cod intern.

Iată secvenţa propusă :

(**B** — conţine iniţial lungimea unui număr **BCD**, adică 5)

```
1   BCDCI1 :    RLD
2               LD (DE),A
3               INC DE
4               XOR A
5               RRD
6               LDI
7               DJNZ BCDCI1
8               LD BC,2
9               LD H,D
10              LD L,E
11              DEC HL
12              LDDR
13              INC HL
14              LD (HL),POINT
15              RET
```

În *bucla principală* a secvenţei (liniile 1—7), întregul număr este despachetat pentru ca apoi partea zecimală să fie deplasată cu o poziţie la dreapta. În golul astfel creat *se inserează punctul zecimal* (liniile 8—13).

Ilustrăm şi această procedură printr-o *schemă sinoptică*. În exemplul din fig. 14.14 *zona sursă* am considerat-o a fi **TMPBCD.**

Listingul comentat al rutinei **BCDCI** se găseşte în Cap. 17, pag. 1—46.

Prezentînd cele trei rutine de conversii a datelor, am parcurs de fapt ramura principală a diagramei de flux informaţional prezentată în Cap. 13.

tastatură ⟶ keybuf ⟶ ebcd ⟶ datbcd ⟶ tmpbcd ⟶ prtbuf ⟶ imprimantă
⟶ afişaj

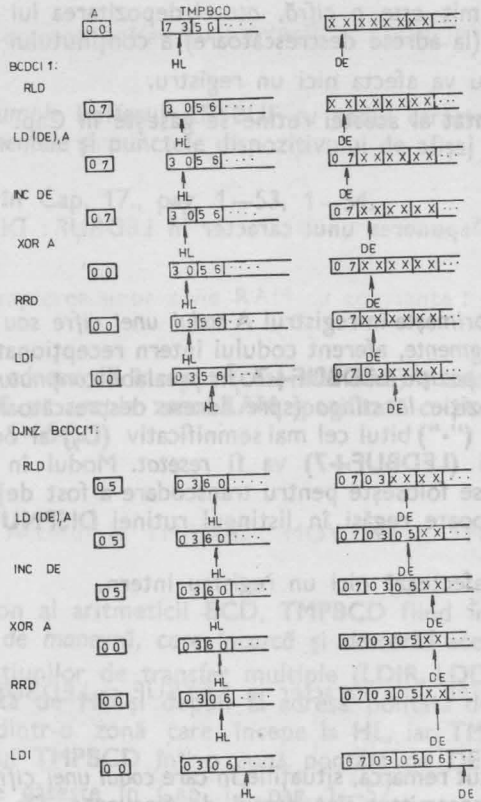


Fig. 14.14. Schema sinoptică de acțiune a rutinei BCDCI

14.5. Vehiculare de date

Rutinele de vehiculare a datelor le-am inclus în ultimul nivel ierarhic. În ceea ce privește complexitatea, rutinele acestei clase acoperă o plajă largă, de la *banalul* **FILLBYTS**, pînă la *interesantul* **EXPAND**.

14.5.1. Depunerea unui caracter în KEYBUF: STORENUM

Rutina **STORENUM** *introduce* în bufferul de tastatură **KEYBUF**, pe poziția **KEYBUF+7**, **cifra** sau **punctul** primit în registrul **A**. **Punctul zecimal** se va concretiza prin *setarea* celui mai semnificativ bit (D_7) al celulei **KEYBUF+7**.

Dacă caracterul primit *este o cifră, atunci* depozitarea lui va fi precedată de o *deplasare la stînga* (la adrese descrescătoare) a conținutului întregului buffer.

STORENUM nu va afecta nici un registru.

Listingul comentat al acestei rutine se găsește în Cap. 17, pag. 1—48.

14.5.2. Depunerea unui caracter în LEDBUF : DISPNUM

Această rutină *primește* în registrul **A** *codul unei cifre sau punct.* Ea va *depune codul de comandă segmente,* aferent codului intern recepționat în bufferul dispozitivului de afișaj, *pe poziția* **LEDBUF+7.** În prealabil *conținutul întregului buffer va fi deplasat* cu o poziție *la stînga* (spre adrese descrescătoare). La recepționarea caracterului **POINT** (**"."**) bitul cel mai semnificativ (D_7) al octetului din extrema dreaptă a afișajului **(LEDBUF+7)** va fi *resetat.* Modul în care generatorul de caractere **LEDGEN** se folosește pentru transcodare a fost deja ilustrat în §14.2.7. **CNTFUNC,** și se poate regăsi în listingul rutinei **DISPNUM** din Cap. 17, pag. 1—49.

DISPNUM nu afectează nici un registru intern.

14.5.3. Depunerea unui caracter în KEYBUF și LEDBUF : STORDISP

Așa cum ați putut remarca, situațiile în care *codul unei cifre sau punct,* trebuie *afișat și memorat concomitent* sînt destul de frecvente. De aceea am constituit rutina **STORDISP,** care apelează pe rînd **STORENUM** și **DISPNUM.**

14.5.4. Normalizarea numerelor introduse : STOREINT

Ca structură, rutina **STOREINT** se aseamănă mult cu **STORNUM.** Ea este apelată din **SYNTAN,** *pentru a normaliza un număr oarecare din* **KEYBUF,** aliniindu-i partea întreagă în dreptul punctului zecimal virtual din **EBCD.**
Listingul comentat al acestei rutine se găsește în Cap. 17 pag. 1—50.

14.5.5. Afișarea unui rezultat din PRTBUF : DISPSUM

Folosind *rutina* **DISPNUM,** rutina pe care o prezentăm *transferă un număr din bufferul de imprimantă în* **LEDBUF.** Ea va fi folosită pentru vizualizarea totalurilor clientului.

Programul sursă se găsește în Cap. 17, pag. 1—51.

14.5.6. Ştergerea bufferelor KEYBUF şi LEDBUF : CLBUFDP

CLBUFDP va *umple* bufferul **KEYBUF** *cu* codul caracterului *spaţiu* (20_H) şi va *stinge toate segmentele* şi punctele dispozitivului de afişaj umplînd **LEDBUF** cu FF_H.

Ea se găseşte în Cap. 17., pag. 1—53, 1—54.

14.5.7. Umplerea unor zone RAM cu constante : FILLBYTS

Primind în **HL** adresa de început (cea inferioară) a unei zone iar în **B** lungimea ei, **FILLBYTS** *va umple zona* **RAM** specificată *cu octetul* din registrul **C.** (vezi Cap. 17, pag. 1—54).

14.5.8. Accesul la TMPBCD : MOVTMP şi TMPMOV

Registrul tampon al aritmeticii **BCD, TMPBCD** fiind frecvent utilizat, s-au creat două *module de manevră, care încarcă* şi *descarcă* acest **registru RAM.**

Aidoma instrucţiunilor de transfer multiple (**LDIR, LDDR,** etc.) care preiau de la adresa indicată de **HL** şi depun la adresa pontată de **DE, MOVTMP** *va încărca* **TMPBCD** dintr-o zonă care începe la **HL,** iar **TMPMOV** *va transfera* conţinutul registrului **TMPBCD** într-o zonă pontată de **DE.**

Listingul lor se găseşte în Cap. 17, pag. 1—52.

14.5.9. Distribuirea parametrilor de stare : TRANSPAR

Nedorind să încheiem Cap. 14 cu banalităţi, am lăsat la urmă două *rutine de manevră* interesante : **TRANSPAR** şi **EXPAND.**

Rutina **TRANSPAR** *transferă* cîte *un parametru de stare,* citit prin procedura **SETUP,** *la locul lui de destinaţie* din **EDBUF.** În momentul apelării, parametrul de transferat se află în **KEYBUF,** *lungimea cuvîntului* de transferat, precum şi *adresa de destinaţie* regăsindu-se *în tabela de distribuire* **SETUPTAB.**

Iată rutina :

```
1  TRANSPAR :   LD   HL,SETUPTAB
2               ADD  HL,BC
3               ADD  HL,BC
4               ADD  HL,BC
5               LD   E,(HL)
6               INC  HL
7               LD   D,(HL)
8               INC  HL
9               LD   B,(HL)
```

```
10                      EX DE,HL
11                      LD DE,KEYBUF+BUFLEN-1
12
12  TAKEBYTE :          LD    A,(DE)
13                      DEC   DE
14                      BIT   7,A
15                      JR    Z,NOPOINT
16                      LD    (HL),POINT
17                      DEC   HL
18                      RES   7,A
19                      DEC   B
20                      JR    Z,ENDTRAN
21  NOPOINT :           LD    (HL),A
22                      DEC   HL
23                      DJNZ  TAKEBYTE
24  ENDTRAN :           RET
    SETUPTAB :          DW    SHOPN
                        DB    SHOPL
                        DW    DESKN
                        .
                        .
                        .
                        DW    CLRKN
                        DB    CLRKL
```

Pe baza numărului de ordine primit în **BC** (valori între 0 și 3) liniile 1—4 *determină adresa intrării* dorite în **SETUPTAB**, de unde *se citește adresa de destinație* și *lungimea* cîmpului util (liniile 6—10).

Bucla de transfer (liniile 12—23) verifică fiecare octet preluat din **KEYBUF** și dacă el conține și un *punct zecimal*, atunci acesta *va fi inserat explicit* (cod pe 1 octet) în cîmpul de destinație (liniile 16—20).

14.5.10. Transferul mesajelor din EPROM în RAM : EXPAND

Sintetizăm acțiunea acestei proceduri printr-o descriere grosieră :

EXPAND

```
procedure expand
  begin
    nr1 : = „numărul liniilor de expandat"
    repeat
      begin
            „transfer din EPROM în RAM, pînă cînd se ajunge a începutul
           rîndului următor"
            „completare cu blancuri"
            nr1 :=nr1-1
      end
    until nr1=0
  end
```

Aceasta fiind ultima descriere în limbaj de nivel înalt cuprinsă în cartea noastră, o vom detailа. EXPAND, pretîndu-se la formalizare, noua ei descriere va reflecta cît se poate de fidel structura programului rezultat în limbaj de asamblare.

Listingul sursă al rutinei EXPAND se găsește în Cap. 17, pag. 1—55.

```
procedure expand
  begin
      nrl :="numărul liniilor de expandat"
      "inițializare rampointer"
      "inițializare rompointer"
      char :=ROM (rompointer)
      repeat
          begin
              counter :="lungimea unei linii"
              repeat
                  begin
                      RAM (rampointer) : = char
                      rompointer : = rompointer+1
                      rampointer : = rampointer+1
                      counter : = counter—1
                      char : = ROM (rompointer)
                  end
              until   char = "începutul rîndului următor"
              repeat
                  begin
                      RAM (rampointer) : = spațiu
                      rampointer : = rampointer+1
                      counter : = counter—1
                  end
              until counter = 0
              nrl : = nrl—1
          end
      until nrl = 0
  end
```

15
MODULE SOFTWARE DEDICATE PENTRU TESTARE

Viața oricărui echipament electronic depinde în primul rînd de fiabilitatea lui, iar în al doilea rînd de ușurința cu care erorile de funcționare pot fi detectate și remediate. Acest din urmă deziderat nu poate fi atins dacă în faza de elaborare a proiectului, aspectului de service-abilitate nu i s-a acordat atenția cuvenită.

Prezența unui microprocesor într-un echipament *poate complica activitatea de service,* ea *putînd solicita* în ultimă instanță utilizarea unor *echipamente de test complexe* cum ar fi, analizorul de stări logice, analizorul de semnături, sau testorul specializat pentru acel produs. Pe teren, inginerului de service, i se oferă rareori posibilitatea utilizării unor astfel de echipamente, fapt care cauzează lungirea timpului mort, cauzat de eventualele defecțiuni ale echipamentului de depanat.

Rememorăm însă faptul că *același microprocesor poate veni în ajutorul personalului de exploatare* și a celui *de service,* datorită faptului că el este *capabil să execute autoteste,* cu condiția ca structura hardware a echipamentului să o permită, și ca modulele program dedicate să fie elaborate și implementate.

Pornind de la considerentele de sus nu vom finaliza proiectul casei de marcat înainte de a fi elaborat module de test și/sau recomandări pentru testarea bunei funcționări a echipamentului. Prezența unui **LED** indicator de „prezență tensiune", acționat pe cale software (vezi §11.3) este un prim pas în acest sens.

„Aprinderea" acestui **LED** este o indicație certă nu numai a faptului că există tensiunile de alimentare cerute ci confirmă și faptul că microprocesorul „lucrează".

Măsura în care bunele noastre intenții se vor concretiza, va reieși din materialul prezentului capitol.

15.1. Test de EPROM

Software-ul rezident al casei de marcat este stocat într-un circuit de memorie de tip **EPROM.** *Alterarea în timp, sau datorită unor avarii, a oricărui* **bit** *din cei 4* **kbyte-ai** *capsulei poate avea efecte inprevizibile.* Există posibilitatea ca alterarea să blocheze total echipamentul, sau să-i provoace un comportament evident

anormal. Caz fericit. Dar ce ne facem dacă eroarea apare în zona rutinelor de aritmetică, şi cauzează în mod aleator erori de calcul ? Comentariile sînt de prisos.

Experienţa demonstrează că *alterarea conţinutului sau a performanţelor electrice statice sau dinamice ale unui* **EPROM,** *nu este exclusă.* De aceea ne vom lua măsuri de precauţie, propunîndu-ne să testăm integritatea informaţiei din **EPROM,** la fiecare iniţializare a echipamentului. *In cazul constatării unei erori, casa de marcat nu va trece în starea operaţională,* tocmai pentru a evita malfuncţionările ascunse.

Pentru semnalizarea unei erori detectate în **EPROM,** am prevăzut (§ 11.3) un indicator luminos care se va putea activa prin bitul D_4 al portului **SYSOUT.**

Ne mai rămîne să stabilim *metoda de testare* a integrităţii informaţiei din **EPROM.** Ideea de bază este de a genera un cuvînt de 1,2 sau 3 octeţi care se calculează din conţinutul **EPROM**-ului, după un algoritm prestabilit, şi de a înscrie şi această informaţie în celulele dedicate din **EPROM.**

Testarea integrităţii conţinutului **EPROM** se poate face executînd o rutină care va *recalcula cuvîntul de control* pe baza informaţiei actuale şi îl *va compara* cu valoarea preînregistrată în locaţiile dedicate. Inegalitatea celor două cuvinte semnalează alterarea informaţiei conţinute în **EPROM.**

Probabilitatea ca apariţia unei sau a mai multor alterări de conţinut să nu modifice cuvîntul de control, şi deci *eroarea să se poată strecura neobservată, depinde de algoritmul* ales pentru constituirea cuvîntului de control şi de lungimea acestuia. Lungirea cuvîntului de control scade probabilitatea nedetectării erorilor, dar poate complica rutina de test, care la rîndul ei trebuie să rezide în **EPROM** scăzînd astfel gradul de eficienţă al utilizării spaţiului **EPROM** (destul de preţios de altfel). Acest conflict se rezolvă după diverse criterii, în funcţie de gradul de securitate impus echipamentului.

În ceea ce priveşte *algoritmii utilizaţi, numărul lor este foarte mare,* oricine putîndu-şi imagina o procedură în acest sens. *Există* cîteva *tehnici consacrate* din care amintim două :

- constituirea unei *sume de control* pe bază de adunări succesive ;
- constituirea unui *cuvînt de control* prin înmulţirea informaţiei cu un polinom de control **(CRC).**

Aceasta din urmă tehnică este des folosită în validarea transferului de date între calculator şi periferice. Ţinînd cont de faptul că echipamentul nostru este unul ipotetic, vom alege algoritmul cel mai simplu.

■ Vom constitui o *sumă de control pe un octet* adunînd toţi octeţii din **EPROM** (exceptînd ultima locaţie) şi neglijînd transportul de la D_7 în sus. Vom calcula complementul faţă de 2 al acestui număr valoare pe care o vom înscrie în ultima locaţie a memoriei **EPROM.** Recalculînd suma întregului **EPROM,** aceasta va trebui să fie **0,** atîta timp cît informaţia cuprinsă rămîne nealterată.

Organigrama unei posibile rutine de test, o redăm în fig. 15.1.

Implementarea algoritmului din fig. 15.1 în limbaj de asamblare nu poate constitui o problemă.

```
EPROMT :    LD      HL,EPROMB       ;iniţializări
            LD      BC,EPROML
            XOR     A
EPROMADD :  ADD     A,(HL)          ;constituirea sumei
            INC     HL
```

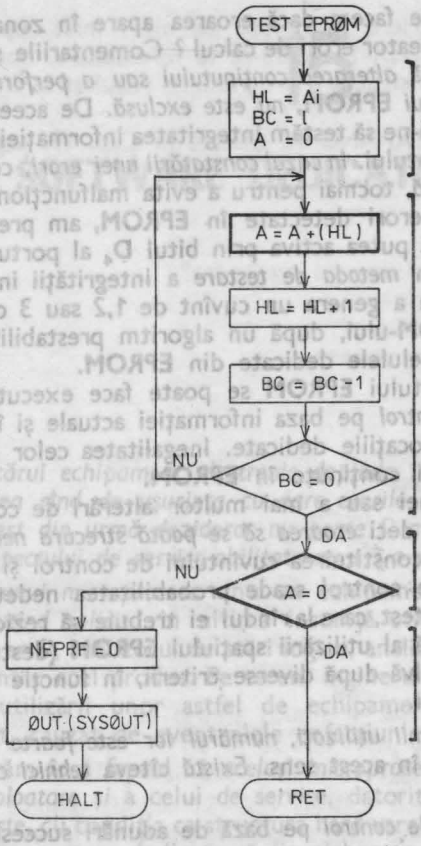


Fig. 15.1. Organigrama rutinei de test conținut EPROM

```
        DEC   BC
        LD    D,A          ;test pentru sfîrșit
        LD    A,B
        OR    C
        LD    A,D
        JR    NZ,EPROMADD
        OR    A            ;verificarea sumei de con-
        JR    Z,OK         ;trol
        LD    A,EPRERMES   ;eventuală semnalizare de
        OUT   (SYSOUT),A   ;eroare
        HALT
OK :    RET
```

In locul instrucțiunii JR Z,OK s-ar fi putut scrie RET Z economisind astfel 2 octeți, dar s-a ales varianta de sus pentru o mai bună corelare cu organigrama din fig. 15.1.

15.2. Test de RAM

Integritatea celulelor de memorie **RAM** *este* la fel de *importantă* ca şi cea a celulelor de **EPROM**, *erorile* cauzate de alterarea informaţiei stocate în ele, *fiind* cel puţin atît de *nefaste* ca şi cele datorate **EPROM**-urilor. *Memoria* **RAM** este acea structură a unui calculator, care *se pretează cel mai bine la testarea „via" program.* Chiar şi în module de memorie constituite din sute de circuite integrate, programe de test adecvate vor putea localiza exact capsula avariată. Această facilitate se datorează faptului că informaţia se poate scrie şi apoi reciti din celulele individuale de memorie **RAM**.

Literatura algoritmilor de test pentru memorie **RAM** *este* deosebit de *bogată*, algoritmii elaboraţi grupîndu-se în jurul a trei deziderate :

- detectarea celulelor de memorie avariate ;
- detectarea erorilor în circuitele de selecţie şi cele de adresare ;
- sesizarea unor erori aleatoare, rare.

În toate cele trei cazuri viteza de execuţie, care în special în cazul urmăririi erorilor rare depinde de algoritmul utilizat, poate fi critică. Dacă spaţiul de memorie de testat ajunge la sute de **kocteţi** sau depăşesc 1 **Mbyte**, parametrul de timp de testare devine deosebit de important.

Noi *vom implementa* în casa de marcat *2 teste* : un **test nedistructiv**, pentru *identificarea* unor *celule* de memorie *avariate* şi altul **distructiv** pentru *verificarea* corectitudinii *circuitelor* de *adresare* şi *selecţie*. În ambele cazuri *distrugerea* se referă la informaţia stocată în prealabil în **RAM** şi nu la circuitul fizic.

15.2.1. Test de RAM nedistructiv (de conţinut)

■ În testul pe care-l propunem, vom verifica şi conţinutul fiecărei celule de memorie în parte salvînd în prealabil şi restaurînd după testare, conţinutul original al celulei testate.

Organigrama unei astfel de rutine o redăm în fig. 15.2.

Vom transpune în limbaj de asamblare algoritmul prezentat în fig. 15.2.

```
RAMT1 :     LD      HL,RAMBOT       ; iniţializări
            LD      BC,RAMLEN
RAMBYTE1 :  LD      A,(HL)          ; salvarea conţinutului
            CPL                     ; iniţial al celulei
            LD      (HL),A          ; înscrierea altei valori
            CP      (HL)            ; test de corectitudine
            JR      NZ,RAMERR
            CPL                     ; restaurarea conţinutului
            LD      (HL),A          ; iniţial al celulei
            INC     HL
            DEC     BC
```

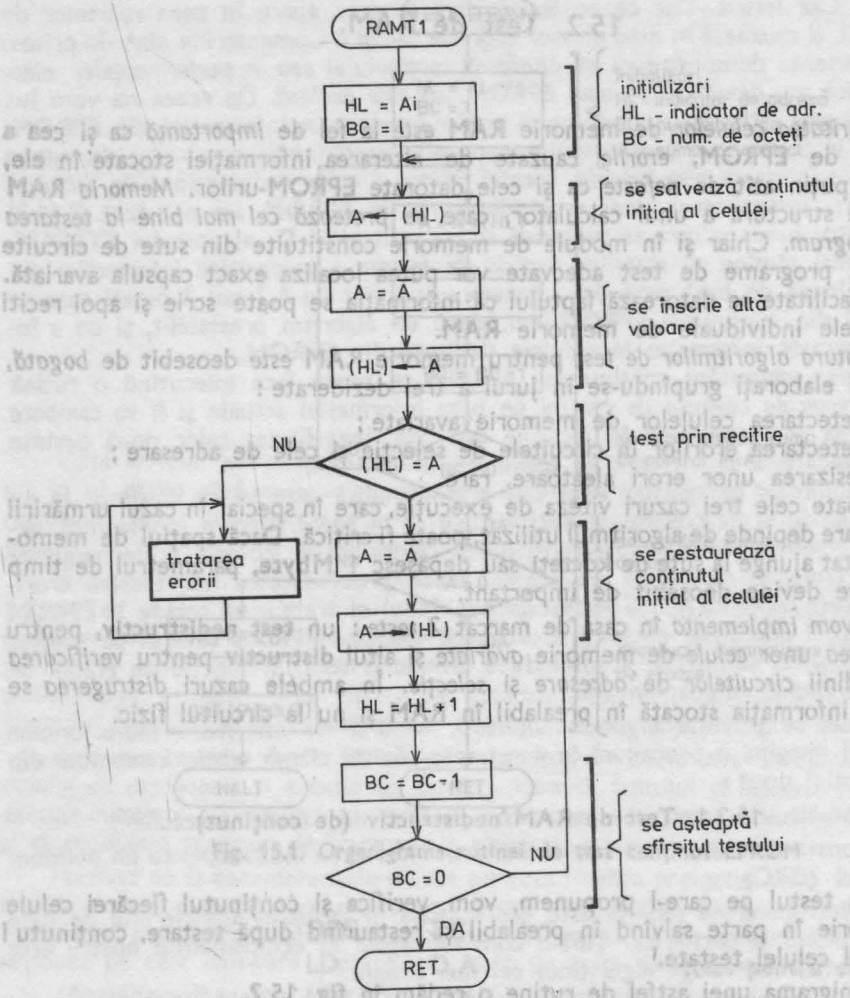


Fig. 15.2. Organigrama testului nedistructiv pentru memoria RAM

```
LD      A,B         ; se așteaptă terminarea tes-
OR      C           ; tului
JR      NZ,RAMBYTE1
RET
```

Testul prezentat detectează erorile de conținut, în schimb nu poate constata eventualele erori de adresare. Ex. pentru două adrese diferite se selectează aceeași celulă. El *are* însă *avantajul nedistructivității*, putînd fi lansat oricînd fără a periclita integritatea datelor conținute în memoria **RAM.** Pentru a verifica și circuitele de adresare vom elabora cel de-al doilea test. Ați remarcat faptul că *ramura de eroare* **(RAMERR)** *este deocamdată netratată.* O vom face după ce vom fi elaborat cel de-al doilea test.

15.2.2. Test de RAM distructiv (de adresare)

■ *Pentru a verifica o* eventuală *suprapunere* a două sau mai multe celule, *testul* de **RAM** *trebuie să decurgă în doi paşi.* În *primul pas se înscrie* întreaga memorie **RAM** disponibilă cu un *conţinut cunoscut.* Valorile înscrise trebuie să difere între ele.

■ *În cel de-al doilea pas se recitește* întreaga zonă **RAM**, *comparînd* fiecare octet cu valoarea înscrisă în primul pas. Dacă apar imperfecţiuni în circuitul de adresare, atunci ele vor putea fi detectate în cel de-al doilea pas.

■ Rămîne de stabilit *algoritmul care să genereze cele* **n** *valori distincte*, care urmează să fie înscrise în memorie pe durata primului pas. Noi *vom folosi* în acest scop însăşi *numărătorul de adresă*, înscriind în grupuri de cîte doi octeţi tocmai adresa primului octet din grup.

Organigrama rutinei **RAMT2** este redată în fig. 15.3.

Codificînd algoritmul, obţinem următorul program:

```
RAMT2  : LD    HL,RAMBOT      ; iniţializări
         LD    BC,RAMLEN
RAMWR  : LD    (HL),L         ; PAS 1
         INC   HL
         LD    (HL),H
         INC   HL
         DEC   BC
         DEC   BC
         LD    A,B
         OR    C
         JR    NZ,RAMWR
         LD    HL,RAMBOT      ; reiniţializări
         LD    BC,RAMLEN
RAMRD  : LD    A,(HL)         ; PAS 2
         CP    L
         JR    NZ,RAMERR
         INC   HL
         LD    A,(HL)
         CP    H
         JR    NZ,RAMERR
         INC   HL
         DEC   BC
         DEC   BC
         LD    A,B
         OR    C
         JR    NZ,RAMRD
         RET
```

● Specificăm faptul că pentru corecta funcţiune a rutinei **RAMT2**, numărul de octeţi de tratat **(RAMLEN)** trebuie să fie un număr par. Din fericire toate circuitele integrate de memorie conţin un număr par de celule.

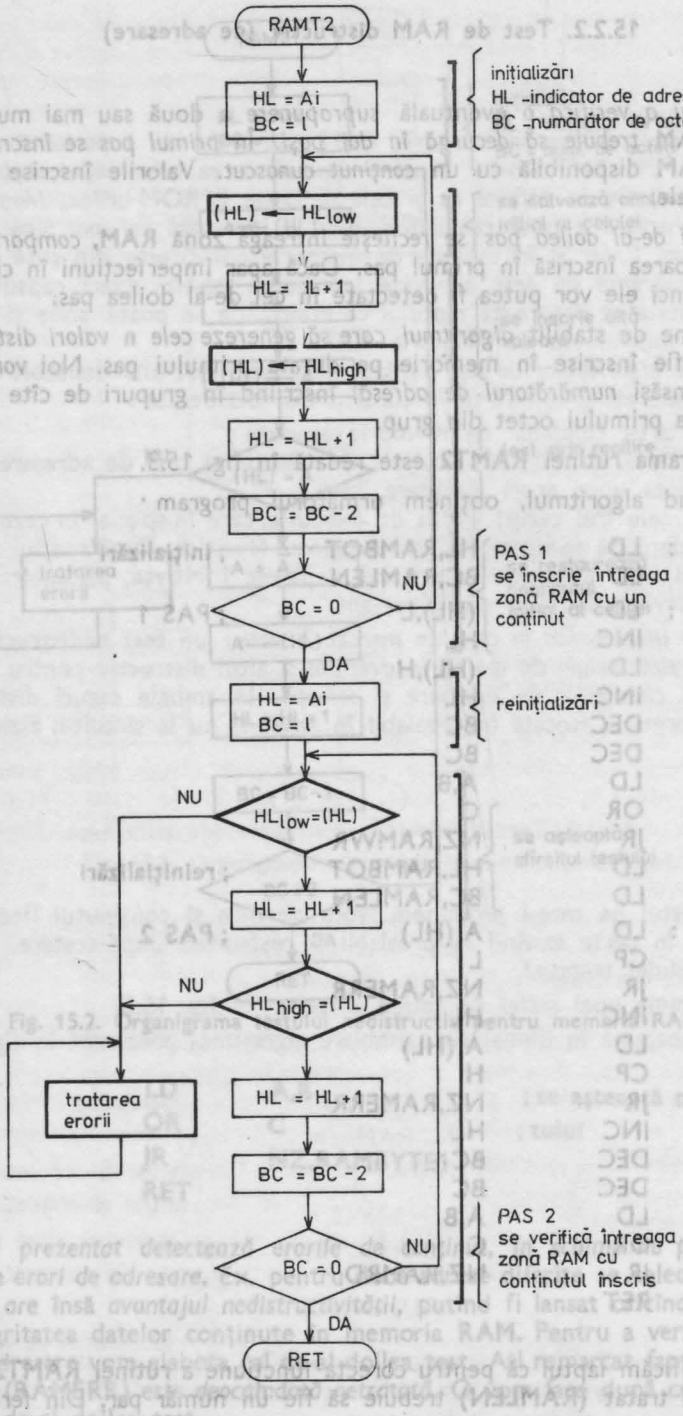


Fig. 15.3. Organigrama testului distructiv pentru memoria RAM

Ca şi în cazul testului de **EPROM**, vom semnala evenimentul activînd un **LED**. În paragraful 11.3 am amintit dioda luminiscentă, care se poate activa prin bitul D_3 al portului de ieşire **SYSOUT (NRAMF)**. *Tratarea* în continuare *a erorilor se va face în mod distinct* de cazul precedent, în care oprirea microprocesorului **(HALT)** s-a dovedit a fi cea mai bună soluţie, căci **EPROM**-ul este în cazul nostru un singur circuit, şi el montat de obicei pe soclu. Personalul de service va putea înlocui cu uşurinţă circuitul integrat EPROM cu altul bun. Circuitele de memorie **RAM** sînt două şi sînt şi lipite. De aceea va trebui să *venim în ajutorul celui care depanează, comunicîndu-i informaţii suplimentare pentru localizarea erorii*. O soluţie ar fi să afişăm pe dispozitivul de afişaj sau pe imprimantă numărul capsulei şi adresa celulei defecte. Dar în cazul echipamentului studiat, ambele interfeţe sînt controlate aproape în totalitate prin software, care presupune funcţionarea ireproşabilă a memoriei **RAM**. Încercînd să afişăm sau să imprimăm ceva, riscăm ca nici una din activităţi să nu aibă efect, sau să genereze mesaje inegale, care vor putea crea confuzii. *Soluţia* care ni se oferă *este de a*

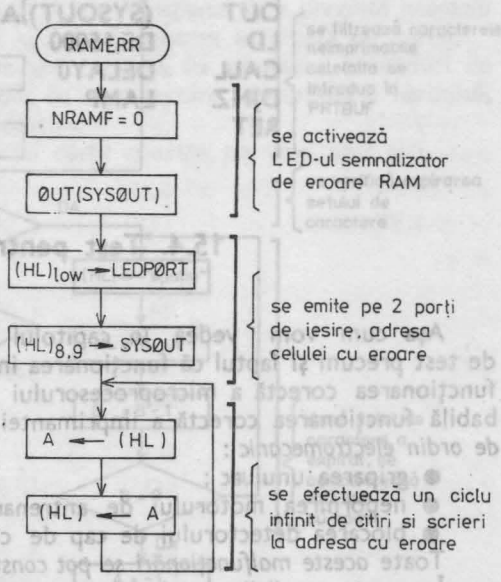


Fig. 15.4. Organigrama secvenţei de tratare a erorilor RAM

emite pe cei 2 porţi de ieşire *adresa la care s-a constatat eroarea*, iar apoi să intrăm într-un ciclu infinit de citire şi scriere a acelei celule. Astfel creăm posibilitatea ca inginerul de service, să detecteze cu ajutorul unui osciloscop cauzele erorii. Organigrama acestei secvenţe de tratare a erorii o redăm în fig. 15.4.

Programul scris în limbajul de asamblare, se regăseşte, completat cu un semnal sonor de avertizare, în listingul din capitolul 17 pag. 1—4, 1—5.

15.3. Test pentru dispozitivul de afişaj

■ Testarea dispozitivului de afişaj *nu se va putea realiza decît cu participarea operatorului, care* cunoscînd ce anume va trebui să se întîmple, *va verifica apariţia evenimentelor preconizate.* În esenţă testul de bună funcţionare a dispozitivului de afişaj constă în *a activa toate segmentele şi punctele zecimale ale dispozitivului,* pentru ca operatorul să le poată verifica „aprinderea". Dacă oricare segment sau punct zecimal este lezat, riscăm să avem indicaţii eronate pe dispozitivul de afişaj.

Iată o secvenţă de test posibilă :

```
LAMPT : LD    B,7
        XOR   A
        OUT   (LEDPORT),A     ; se activează toate segmentele
                              ; şi punctul zecimal
LAMP  : LD    A,(WITNESS)
        AND   MASK2
        OR    B
        LD    (WITNESS),A     ;
        OUT   (SYSOUT),A      ; se selectează unul din
        LD    DE,15000        ; cele opt elemente de afişaj
        CALL  DELAY0          ; se aşteaptă cca. 0,3 s.
        DJNZ  LAMP            ; se comută pe următorul LED
        RET
```

15.4. Test pentru imprimantă

Aşa cum vom vedea în capitolul următor, ordinea de apel a rutinelor de test precum şi faptul că funcţionarea imprimantei depinde în mare măsură de funcţionarea corectă a microprocesorului şi de integritatea memoriei, fac probabilă funcţionarea corectă a imprimantei. În acest caz *erorile posibile sînt cele de ordin electromecanic* :

- griparea unui ac ;
- nepornirea motorului de antrenare, sau blocarea camei de transport ;
- blocarea detectorului de cap de cursă.

Toate *aceste malfuncţionări se pot constata la prima vedere.*

În aceste condiţii *prevederea unui test dedicat imprimantei poate fi redundantă.* De aceea nici nu o vom include în proiectul final. Respectînd prevederile specificaţiei funcţionale **F.1.0.**, operatorul va avea siguranţa bunei funcţionări a imprimantei înainte de a emite primul bon client în sesiunea respectivă de lucru.

Ca fapt divers *amintim totuşi că majoritatea imprimantelor de sine stătătoare sînt prevăzute cu programe de test,* care listează rînduri complete conţinînd setul de caractere imprimabile.

În fig. 15.5. redăm organigrama unui astfel de test pentru casa de marcat considerată.

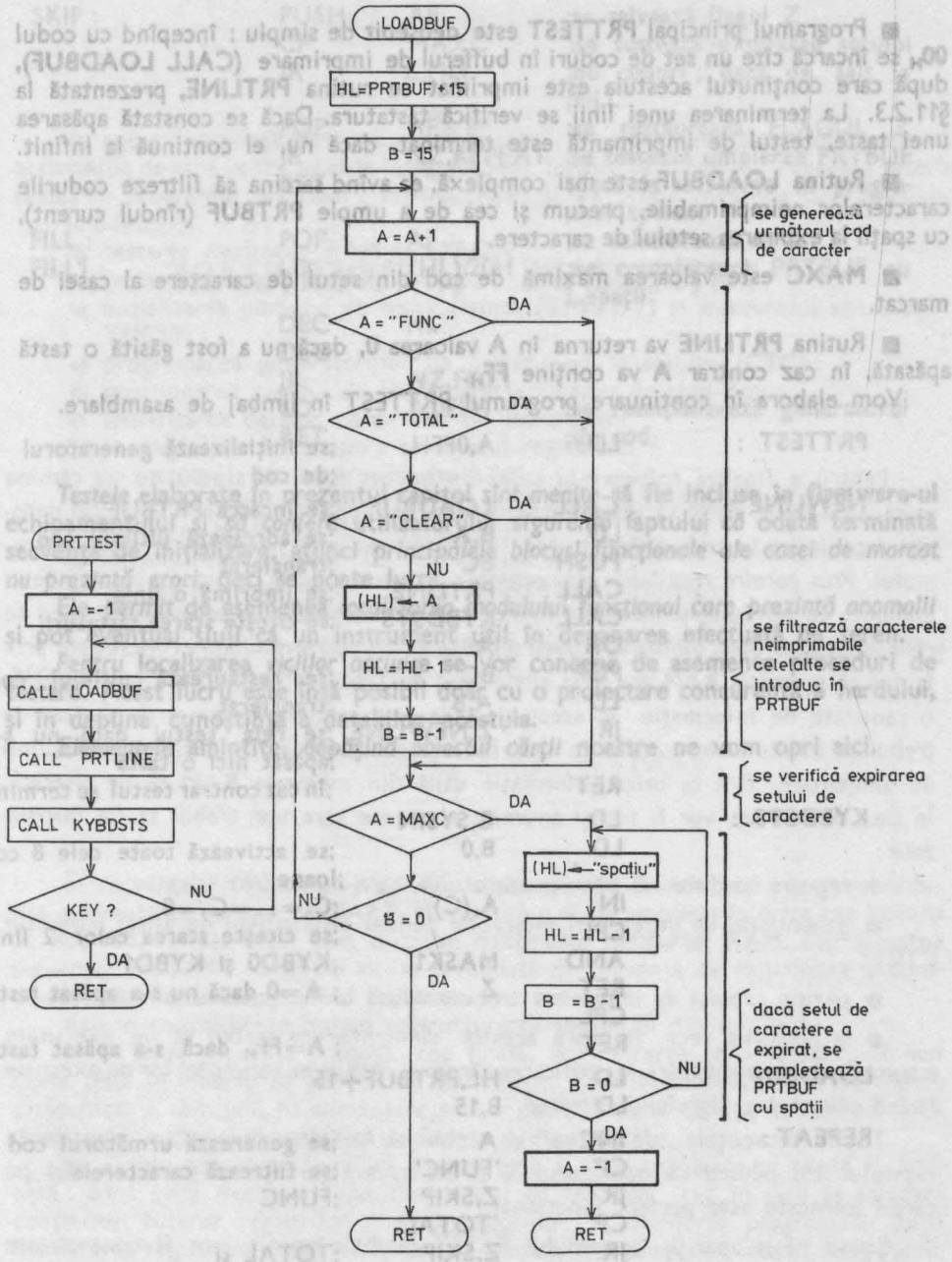


Fig. 15.5. Test pentru imprimantă : PRTTEST

■ Programul principal **PRTTEST** este deosebit de simplu : începînd cu codul 00_H se încarcă cîte un set de coduri în bufferul de imprimare **(CALL LOADBUF)**, după care conținutul acestuia este imprimat cu rutina **PRTLINE**, prezentată la §11.2.3. La terminarea unei linii se verifică tastatura. Dacă se constată apăsarea unei taste, testul de imprimantă este terminat, dacă nu, el continuă la infinit.

■ Rutina **LOADBUF** este mai complexă, ea avînd sarcina să filtreze codurile caracterelor neimprimabile, precum și cea de a umple **PRTBUF** (rîndul curent), cu spații la expirarea setului de caractere.

■ **MAXC** este valoarea maximă de cod din setul de caractere al casei de marcat.

■ Rutina **PRTLINE** va returna în **A** valoarea **0**, dacă nu a fost găsită o tastă apăsată, în caz contrar **A** va conține FF_H.

Vom elabora în continuare programul PRTTEST în limbaj de asamblare.

```
PRTTEST :   LD      A,0FFH      ;se inițializează generatorul
                                ;de cod
NEWLINE :   CALL    LOADBUF     ;se încarcă PRTBUF
            LD      B,A         ;se adresează ultimul cod
            PUSH    BC          ;transferat
            CALL    PRTLINE     ;se imprimă o linie
            CALL    KYBDSTS     ;se citește starea tastaturii
            OR      A
            POP     BC          ;se restaurează ultimul cod
            LD      A,B         ;transferat
            JR      Z,NEWLINE   ;se reia testul dacă nu s-a
                                ;apăsat nici o tastă
            RET                 ;în caz contrar testul se termină
KYBDSTS :   LD      C,SYSIN
            LD      B,0         ;se activează toate cele 8 co-
                                ;loane
            IN      A,(C)       ;C0=..=C7=0
            CPL                 ;se citește starea celor 2 linii
            AND     MASK1       ;KYBD0 și KYBD1
            RET     Z           ; A=0 dacă nu s-a apăsat tastă
            CPL
            RET                 ; A=FFH dacă s-a apăsat tastă
LOADBUF :   LD      HL,PRTBUF+15
            LD      B,15
REPEAT :    INC     A           ;se generează următorul cod
            CP      'FUNC'      ;se filtrează caracterele
            JR      Z,SKIP      ;FUNC
            CP      'TOTAL'
            JR      Z,SKIP      ;TOTAL și
            CP      'CLEAR'
            JR      Z,SKIP      ;CLEAR
            LD      (HL),A      ;orice alt cod se introduce în
            DEC     HL          ;PRTBUF
            DEC     B
```

```
SKIP :    PUSH   AF          ;se salvează flagul Z
          CP     MAXC        ;se testează expirarea setului
          JR     Z,FILL      ;de coduri ; dacă da, salt la
                             ;FILL
          POP    AF          ;se restaurează FLAGUL Z
          JR     NZ,REPEAT   ;se testează umplerea PRTBUF
          RET                ;dacă da se revine în progra-
                             ;mul apelant
FILL :    POP    AF          ;se echilibrează stiva
FILL1 :   LD     (HL),20H    ;se completează PRTBUF cu
                             ;„spații"
          DEC    HL
          DEC    B
          JR     NZ,FILL1
          LD     A,0FFH      ;se reinițializează generatorul
          RET                ;de cod
```

Testele elaborate în prezentul capitol *sînt menite* să fie incluse în firmware-ul echipamentului și *să confere* utilizatorului *siguranța* faptului *că* odată terminată secvența de inițializare, atunci *principalele blocuri funcționale ale casei de marcat nu prezintă erori*, deci se poate lucra.

Ele *permit* de asemenea *localizarea modulului funcțional care prezintă anomalii* și pot eventual sluji ca un instrument util în depanarea efectuată pe teren.

Pentru localizarea *viciilor ascunse* se vor concepe de asemenea proceduri de testare. Acest lucru este însă posibil doar cu o proiectare concurentă a hardului, și în deplina cunoștință a detaliilor acestuia.

Elementele amintite, *depășind obiectul cărții* noastre ne vom opri aici.

16

INIȚIALIZARE ȘI SUPERVIZARE

Pentru a finaliza software-ul casei de marcat luată în studiu nu ne rămîne altceva de făcut, decît să *elaborăm secvențele de inițializare*. Vorbim la plural datorită faptului că va trebui să *distingem* între **pornirea rece** și **pornirea caldă** a sistemului. *Prin pornire rece înțelegem cuplarea la rețea a casei*, moment în care se va alimenta cu tensiune și memoria **RAM**. *Pornirea caldă se va efectua* atunci cînd se reia activitatea casei de marcat, *după o pauză de curent*, eveniment concretizat prin faptul că memoria **RAM** era deja alimentată cu tensiune, în ea fiind înscrisă deja o cantitate de informație. În acest al doilea caz, activitatea casei de marcat va trebui să continue în punctul în care ea fusese abandonată la dispariția tensiunii de alimentare, fără ca orice informație utilă din memoria **RAM** să fie alterată. În casa de marcat, vor fi totuși anumite elemente care vor trebui să fie inițializate :

- *registri hardware* ai microprocesorului, care nu au fost salvați ;
- *generatorul de întreruperi* mascabile pentru dispozitivul de afișaj (circuitul CTC) ;
- *poziția capului de imprimare* trebuie adusă la începutul unui rînd nou ;
- la pornirea rece, în afara acestor elemente vor trebui să fie executate o sumedenie de inițializări, identificarea cărora și stabilirea secvenței lor de execuție făcînd obiectul paragrafului următor.

Tot în secvențele *„de trezire"* va trebui să *includem și testele* prezentate în capitolul 15, pentru ca operatorul să poată avea siguranța că echipamentul pe care-l folosește este perfect funcțional.

Știind că la apariția semnalului **RESET** (ambele porniri vor fi concretizate prin apariția acestui semnal) execuția programului începe la adresa **0**, secvențele de inițializare vor trebui dispuse la începutul **EPROM**-ului. *Cele 2 secvențe de trezire avînd* astfel în mod obligatoriu *o parte comună, va trebui să decidem* în mod judicios *momentul în care ele se vor despărți*, urmîndu-și calea pentru a deveni evenimente distincte : *pornire* (start) sau *repornire* (restart).

16.1. Pornirea rece : CSTART

În această secvență pornim cu *„tabula rasa"*. Ne propunem să trecem în revistă activitățile pe care va trebui să le întreprindem, înainte de a intra în repaosul interbon (adică în programul principal : **MAINLOOP**) și a-i preda comanda operatorului. Iată-le :

- testarea memoriei **RAM** și **EPROM** ;
- testarea dispozitivului de afișaj ;
- inițializarea portului de ieșire sistem **(SYSOUT)** și martorului său **(WITTNESS)** ;
- programarea generatorului de întreruperi mascabile **(CTC)** ;
- poziționarea capului de imprimare la început de rînd nou ;
- specificarea datelor sistemului de întreruperi
  - — modul de întrerupere și octetul registrului I
  - — vectorul de întreruperi în **RAM** ;
- transferul mesajelor din **EPROM** în **EDBUF (EXPAND)** ;
- ștergerea bufferelor de intrare/ieșire ;
- inițializarea celulelor ce memorează vînzări ;
- înscrierea unei cifre **"0"** pe dispozitivul de afișaj ;
- validarea sistemului de întreruperi.

Astfel se constituie organigrama din fig. 16.1 care este o primă aproximație a procedurii de pornire rece a casei de marcat.

16.2. Pornirea caldă : WSTART

În secvența din Fig. 16.1 n-am marcat punctul de decizie în care se va ramifica secvența de pornire **WSTART**. El va trebui să fie amplasat în orice caz înainte de **RAMT2**, fiindcă acest test distruge conținutul memoriei **RAM**. Dar lansarea secvenței **WSTART** va trebui să fie precedată de secvența de inițializare a componentelor hardware căci ele au rămas fără tensiune.

Aceste considerente impun restructurarea secvenței din fig. 16.1.

■ În noua secvență propusă, cea finală, vom încerca să amplasăm *cît mai multe teste în amonte de punctul de ramificație*, datorită faptului că orice cădere accidentală a tensiunii de alimentare este o posibilă sursă de avarii. La repornirea casei, principalele blocuri funcționale vor trebui testate.

■ Aceste considerente ridică o problemă a cărei rezolvare nu poate fi amînată : *dacă între momentul apariției căderii de tensiune*, caz în care se salvează conținutul tuturor regiștrilor microprocesorului, *și relansarea programului* după restaurarea regiștrilor **WSTART**, *se vor executa alte programe, atunci integritatea memoriei* **RAM** *va fi periclitată* de însăși apelurile de subrutine sau salvările de regiștri, care vor scrie pe stivă, în secvența de trezire care precede punctul de ramificație **CSTART, WSTART.**

■ Pentru a fi siguri că aceste manevre pe stivă nu vor afecta cu nimic conținutul **RAM**, *vom rezerva o zonă de stivă dedicată amplasată în capătul memoriei*, zonă care va fi *folosită în exclusivitate de secvența comună de trezire*. După punctul

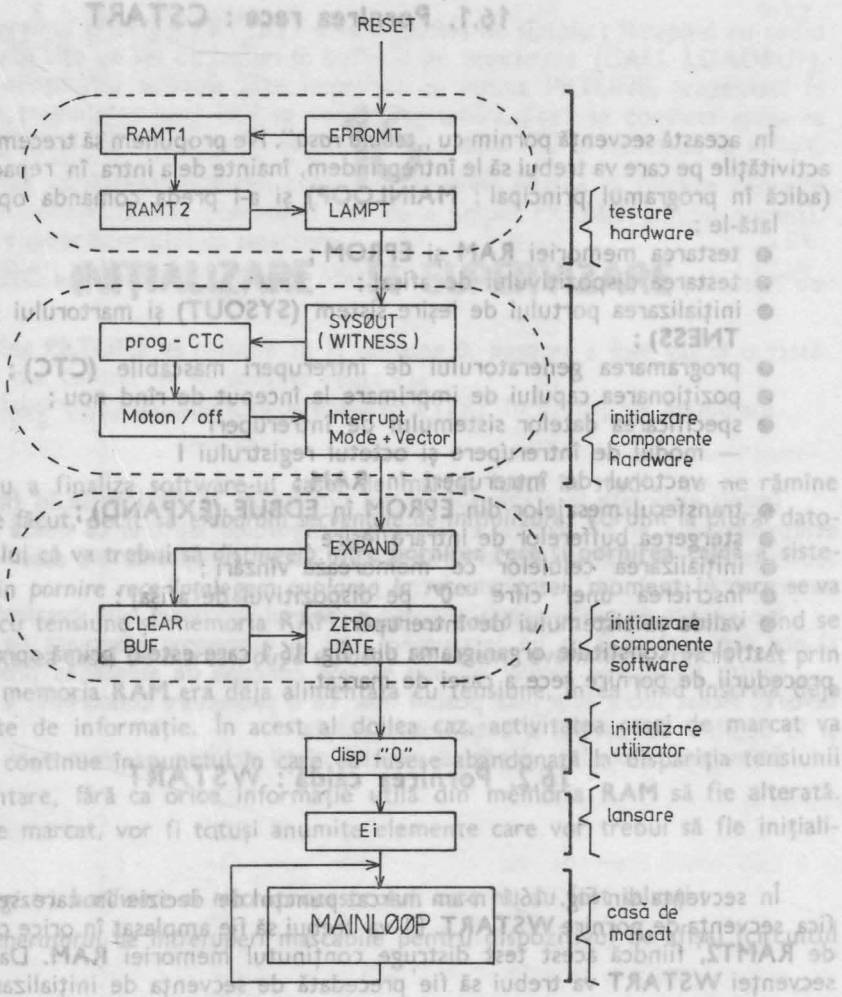


Fig. 16.1. Secvența pornirii reci (o primă aproximație)

de ramificație, **CSTART** își va inițializa indicatorul de stivă **SP** la o adresă inferioară zonei rezervate pentru secvența comună, iar **WSTART** va prelua noua valoare pentru **SP** din celulele **RAM** în care ea fusese salvată la căderea tensiunii (a se vedea rutina **DROPOUT**). *Lungimea zonei rezervate* o vom stabili după definitivarea secvențelor de trezire, urmată de o analiză minuțioasă a modului în care secvența care precede punctul de ramificație va utiliza stiva.

În fig. 16.2. redăm organigrama finală a secvențelor de pornire : (a se vedea specificația funcțională **F.0.19.** și **F.1.0.**)

■ *Ramificația* **CSTART, WSTART** se va face pe baza comparării primei linii (9 octeți) din **EDBUF** cu primii 9 octeți ai tabelei de mesaje **(MESSAGE)** din **EPROM**, așa cum am amintit deja în capitolul 13. Știm că **EDBUF** se generează executînd rutina **EXPAND**, la trezirea rece a casei de marcat. Dacă pe urma com-

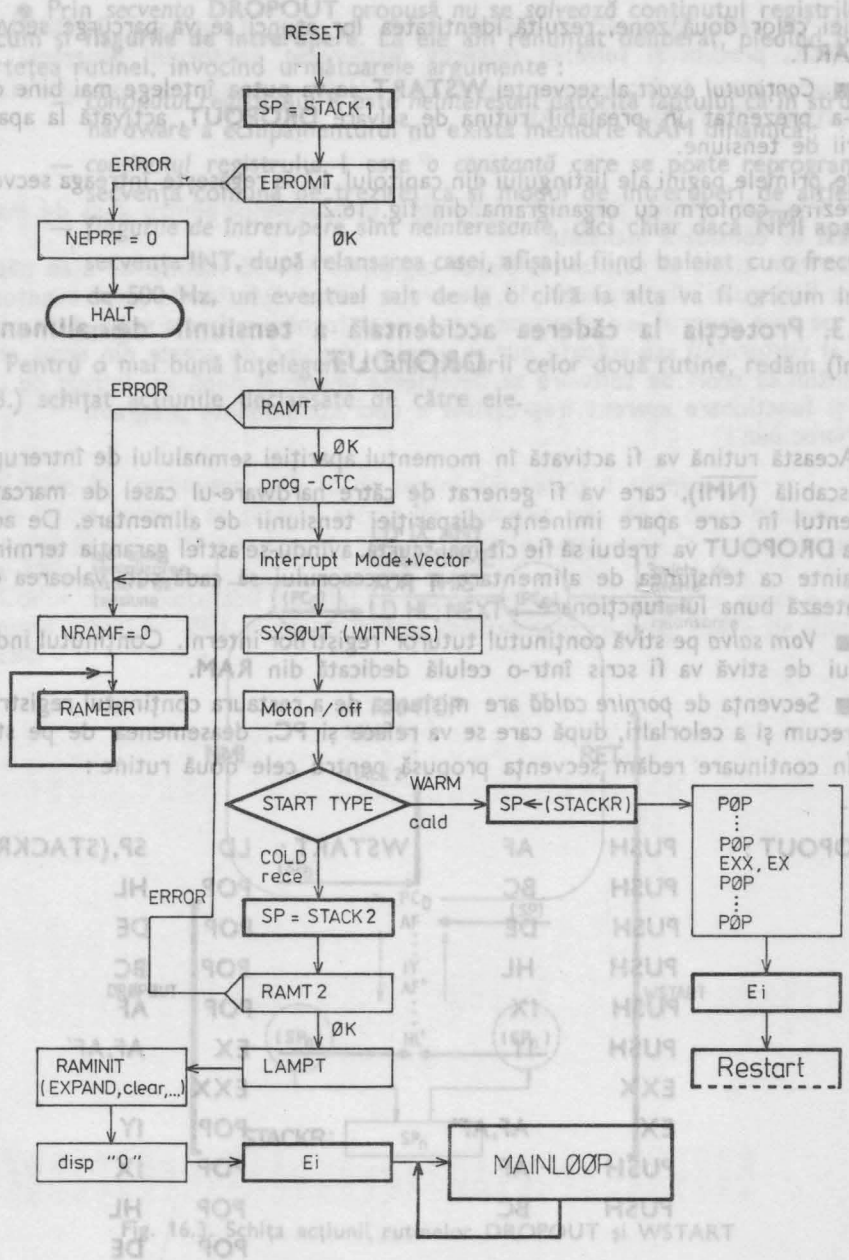


Fig. 16.2. Organigrama secvenţelor de pornire (aproximaţia finală)

paraţiei celor două zone, rezultă identitatea lor atunci se va parcurge secvenţa **WSTART.**

■ *Conţinutul exact* al secvenţei **WSTART** se va putea înţelege mai bine după ce s-a prezentat în prealabil rutina de salvare **DROPOUT,** activată la apariţia căderii de tensiune.

Pe primele pagini ale listingului din capitolul 17 se regăseşte întreaga secvenţă de trezire, conform cu organigrama din fig. 16.2.

16.3. Protecţia la căderea accidentală a tensiunii de alimentare : DROPOUT

Această rutină va fi activată în momentul apariţiei semnalului de întrerupere nemascabilă **(NMI),** care va fi generat de către hardware-ul casei de marcat în momentul în care apare iminenţa dispariţiei tensiunii de alimentare. De aceea rutina **DROPOUT** va trebui să fie cît mai scurtă, avîndu-se astfel garanţia terminării ei înainte ca tensiunea de alimentare a procesorului să cadă sub valoarea care garantează buna lui funcţionare.

■ *Vom salva* pe stivă conţinutul tuturor regiştrilor interni. Conţinutul indicatorului de stivă va fi scris într-o celulă dedicată din **RAM.**

■ Secvenţa de *pornire caldă* are misiunea de a restaura conţinutul registrului SP precum şi a celorlalţi, după care se va reface şi **PC,** deasemenea de pe stivă.

În continuare redăm secvenţa propusă pentru cele două rutine :

```
DROPOUT :  PUSH   AF          WSTART :  LD    SP,(STACKR)
           PUSH   BC                    POP   HL
           PUSH   DE                    POP   DE
           PUSH   HL                    POP   BC
           PUSH   IX                    POP   AF
           PUSH   IY                    EX    AF,AF'
           EXX                          EXX
           EX     AF,AF'                POP   IY
           PUSH   AF                    POP   IX
           PUSH   BC                    POP   HL
                                        POP   DE
           PUSH   DE                    POP   BC
           PUSH   HL                    POP   AF
                                        EI
           LD     (STACKR),SP           RET
           HALT
```

● Prin *secvența* **DROPOUT** propusă *nu* se *salvează* conținutul regiștrilor **R**, precum și **flagurile de întrerupere**. La ele am renunțat deliberat, pledînd pentru scurtețea rutinei, invocînd următoarele argumente :

— *conținutul* registrului **R** este *neinteresant* datorită faptului că în structura hardware a echipamentului nu există memorie **RAM** dinamică ;

— *conținutul* registrului **I** este *o constantă* care se poate reprograma în secvența comună de trezire, ca și modul de întreruperi de altfel ;

— *flagurile de întrerupere* sînt *neinteresante*, căci chiar dacă $\overline{\text{NMI}}$ apare în secvența **INT**, după relansarea casei, afișajul fiind baleiat cu o frecvență de 500 **Hz**, un eventual salt de la o cifră la alta va fi oricum insesizabil.

Pentru o mai bună înțelegere a funcționării celor două rutine, redăm (în fig. 16.3.) schițat acțiunile declanșate de către ele.

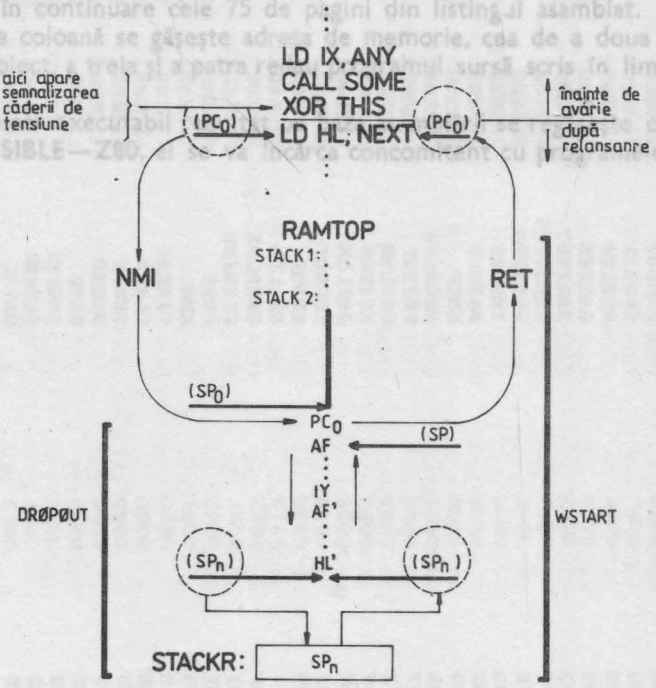


Fig. 16.3. Schița acțiunii rutinelor DROPOUT și WSTART

● Din reprezentarea grafică sperăm să rezulte cît se poate de clar *modul în care se abortează* secvența considerată ca exemplificare (după execuția instrucțiunii **XOR THIS**) și *cum se reia „firul pierdut"* începînd cu instrucțiunea **LD HL,NEXT.**

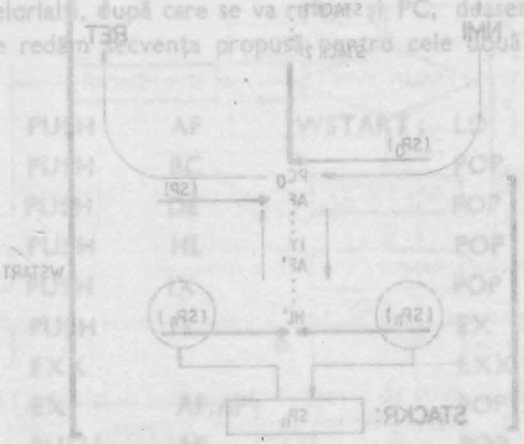
● Va trebui să *observăm evoluţia stivei*, ca efect al rutinelor **DROPOUT** şi **WSTART**, precum şi *salvarea respectiv restaurarea* **contorului program**, prin acceptarea cererii de întrerupere nemascabilă **(NMI)** respectiv prin execuţia ultimei instrucţiuni **(RET)**, din **WSTART**.

*

Cu aceste ultime precizări elaborarea software-ului pentru casa de marcat virtuală se consideră încheiată.

Invităm cititorul să mai răsfoiască capitolele 12—16 înainte de a se năpusti asupra listingului din capitolul 17, pentru a putea detecta mai uşor eventualele erori pe care autorul le-a strecurat intenţionat (pentru a forma spiritul de observaţie al cititorului) sau scăpate pur şi simplu. Vorbind de aceste din urmă erori, reamintim că erori de software se detectează uneori şi după ani întregi de utilizare şi funcţionare aparent ireproşabilă a unei componente program.

Noroc bun !



17

LISTA COMENTATĂ A PROGRAMULUI

Redăm în continuare cele 75 de pagini din listingul asamblat.

În prima coloană se găsește adresa de memorie, cea de a doua coloană conține codul obiect, a treia și a patra redau programul sursă scris în limbaj de asamblare **Z80.**

Codul binar executabil rezultat pe baza asamblării se regăsește octet cu octet pe caseta **VISIBLE—Z80,** el se va încărca concomitent cu programele de simulare **VISZ80.**

```
MACRO-80 5.80   15-Jun-86       PAGE    1-1


                        ;       Initializarea variabilelor utilizate
                        ;
6C00                    RAMBOT          EQU     6C00H
0400                    RAMLEN          EQU     400H
0005                    BCDLEN          EQU     5
6C00                    DAYBCD          EQU     RAMBOT
6C05                    SORTTOT         EQU     DAYBCD+BCDLEN
6DF9                    GUESTBCD        EQU     SORTTOT+BCDLEN*100
6DFE                    WORKBCD         EQU     GUESTBCD+BCDLEN
6E03                    TMPBCD          EQU     WORKBCD+BCDLEN
6E08                    DATBCD          EQU     TMPBCD+BCDLEN
6E0D                    EBCD            EQU     DATBCD+BCDLEN
000A                    EBCDLEN         EQU     10
6E17                    PRTBUF          EQU     EBCD+EBCDLEN
0012                    PRTLEN          EQU     18
6E29                    KEYBUF          EQU     PRTBUF+PRTLEN
0008                    BUFLEN          EQU     8
6E31                    LEDBUF          EQU     KEYBUF+BUFLEN
6E39                    EDBUF           EQU     LEDBUF+BUFLEN
0011                    EDLLEN          EQU     17
000E                    NREDLNS         EQU     14
6F27                    WITNESS         EQU     EDBUF+EDLLEN*NREDLNS
6E57                    CODE            EQU     EDBUF+EDLLEN*2-4
6E93                    SUM             EQU     EDBUF+EDLLEN*5+5
6EB3                    TICKNR          EQU     EDBUF+EDLLEN*7+3
6EBD                    CLRKN           EQU     TICKNR+10
6EF0                    SHOPN           EQU     EDBUF+EDLLEN*11-4
6F01                    DESKN           EQU     EDBUF+EDLLEN*12-4
6F0A                    DATE            EQU     EDBUF+EDLLEN*12+5
0003                    SHOPNL          EQU     3
0003                    CLRKNL          EQU     3
0008                    DATEL           EQU     8
0002                    DESKNL          EQU     2
0009                    CONTRLEN        EQU     9
6F28                    STACKR          EQU     WITNESS+1
6FF8                    STACK1          EQU     6FF8H
6FF0                    STACK2          EQU     STACK1-8
6FF8                    INTTAB          EQU     STACK1
```

```
                6F2A    NRFUNC  EQU     STACKR+2
                0020    BLANK   EQU     20H
                000D    ASTER   EQU     0DH
                0010    ANULARE EQU     10H
                0097    CTCCMD  EQU     10010111B
                00FA    CTCCOUNT EQU    250
                0090    SYSOUT  EQU     90H
                0090    SYSIN   EQU     90H
                00A0    LEDPORT EQU     0A0H
                00B0    PRTPORT EQU     0B0H
                00C0    CTC     EQU     0C0H
                0001    KEY1    EQU     1
                0002    KEY2    EQU     2
                00AF    EPRERMES EQU    10101111B
                00B7    RAMEMES EQU     10110111B
                2710    RAMERHFR EQU    10000
                03E8    RAMERLFR EQU    1000
                00C8    RAMERTM EQU     200
                000C    KYBDMASK EQU    00001100B
                0008    BUYMASK EQU     8
                0004    SORTMASK EQU    4
                0002    DAYMASK EQU     2
                0001    SYNTMASK EQU    1
                000B    FUNC    EQU     0BH
                000B    NRORPT  EQU     FUNC
                000E    TOTAL   EQU     0EH
                000C    PLUS    EQU     0CH
                0022    MINUS   EQU     22H
                000F    CLEAR   EQU     0FH
                000A    POINT   EQU     0AH
                00FF    BLKDISP EQU     0FFH
                7000    EPROMB  EQU     7000H
                1000    EPROML  EQU     1000H
                0007    FUNCNUM EQU     7
                0002    MAXFUNC2 EQU    2
                0002    SCLEN   EQU     2
                0004    TCKNRLEN EQU    4
                00BF    NORMSOUT EQU    10111111B
                0007    BEEPBIT EQU     7
                0BB8    FCSDFRE EQU     3000
```

```
MACRO-80 5.80   15-Jun-86  PAGE    1-3

0014                    FCSDTIME        EQU     20
0320                    SENDFR          EQU     800
0014                    SENDTM          EQU     20
03E8                    OPERFR          EQU     1000
0014                    OPERTM          EQU     20
03E8                    EDBFRE          EQU     1000
0014                    EDBTIME         EQU     20
07D0                    ERSDFR          EQU     2000
0014                    ERSDTM          EQU     20
0190                    TCLOCK          EQU     400     ; nanosecunde
61A8                    ONEHZ           EQU     25000
0070                    INERTDEL        EQU     70H
0007                    CARLIMIT        EQU     7
0080                    BTWTDEL         EQU     80H
0005                    BTWCHDEL        EQU     5
0001                    SETDEL          EQU     1
0004                    RESDEL          EQU     4
0035                    BASEDEL         EQU     53
0080                    RESNEAD         EQU     80H
0080                    MOTSTART        EQU     80H
0000                    MOTSTOP         EQU     0
000C                    MASK1           EQU     00001100B
00F8                    MASK2           EQU     11111000B
0003                    SHIFTM          EQU     3
7000                    START:
7000    31 6FF8             LD      SP,STACK1       ; se initializeaza stiva pentru secventa de
                                                    ; pornire
                        ;
                        ;   Test de EPROM
                        ;
7003                    EPROMT1:
7003    21 7000             LD      HL,EPROMB       ; in HL adresa de inceput a memoriei EPROM
7006    01 1000             LD      BC,EPROML       ; in BC numarul de octeti de testat
7009    AF                  XOR     A               ; A = 0
700A                    EPROMADD:
700A    86                  ADD     A,(HL)          ; octetul curent se aduna la A si
700B    23                  INC     HL              ; se trece la urmatorul octet
700C    0B                  DEC     BC              ; se decrementeaza numaratorul.
700D    57                  LD      D,A             ; se salveaza totalul curent si
700E    78                  LD      A,B             ; se testeaza daca BC = 0 ?
```

```
700F  B1                    OR      C
7010  7A                    LD      A,D             ; se reface totalul ( indicatorul Z neafectat )
7011  20 F7                 JR      NZ,EPROMADD     ; daca BC nu este 0,salt inapoi
7013  B7                    OR      A               ; se pozitioneaza indicatorul Z
7014  28 05                 JR      Z,RAMT1         ; daca A = 0,salt la testul de RAM nr.1
7016                  EPROMERR:                     ; daca nu,
7016  3E AF                 LD      A,EPRERMES      ; se trimite mesaj de eroare la portul SYSOUT
7018  D3 90                 OUT     (SYSOUT),A      ; ( NEPRF = 0 )
701A  76                    HALT
                      ;
                      ;     Test de RAM nr. 1
                      ;
701B                  RAMT1:
701B  21 6C00               LD      HL,RAMBOT       ; in HL adresa de inceput a memoriei RAM
701E  01 0400               LD      BC,RAMLEN       ; in BC numarul de octeti de testat
7021                  RAMBYTE1:
7021  7E                    LD      A,(HL)          ; se trece octetul curent in A
7022  2F                    CPL                     ; se complementeaza
7023  77                    LD      (HL),A          ; se rescrie si
7024  BE                    CP      (HL)            ; se testeaza daca s-a rescris corect
7025  20 0A                 JR      NZ,RAMERR       ; daca nu,salt la eroare RAM
7027  2F                    CPL                     ; daca a fost corect,se reface in A valoarea
7028  77                    LD      (HL),A          ; initiala si se rescrie in memorie
7029  23                    INC     HL              ; se trece la urmatorul octet
702A  0B                    DEC     BC              ; se decrementeaza numaratorul si
702B  78                    LD      A,B             ; se testeaza daca a ajuns la 0
702C  B1                    OR      C
702D  20 F2                 JR      NZ,RAMBYTE1     ; daca nu,salt inapoi
702F  18 2E                 JR      CTCPROG         ; daca da,salt la programarea circuitului CTC
7031                  RAMERR:
7031  3E B7                 LD      A,RAMEMES       ; se trimite mesaj de eroare la portul SYSOUT
7033  D3 90                 OUT     (SYSOUT),A      ; ( NRAMF = 0 )
7035  54                    LD      D,H             ; adresa la care s-a detectat eroarea se
7036  5D                    LD      E,L             ; salveaza in DE
7037                  RAMESD:
7037  21 2710               LD      HL,RAMERHFR     ; se suna soneria la frecventa inalta
703A  3E C8                 LD      A,RAMERTM
703C  CD 77C3               CALL    BEEP
703F  21 03E8               LD      HL,RAMERLFR     ; se suna soneria la frecventa joasa
7042  3E C8                 LD      A,RAMERTM
```

```
MACRO-80 5.80   15-Jun-86        PAGE    1-5

7044   CD 77C3                   CALL    BEEP
7047   0E 90                     LD      C,SYSIN         ; se citeste portul de intrari SYSIN
7049   06 00                     LD      B,0
704B   ED 78                     IN      A,(C)
704D   E6 0C                     AND     KYBDMASK        ; si se testeaza starea tastaturii
704F   20 E6                     JR      NZ,RAMESD       ; daca nu este tasta apasata,salt inapoi
7051                    WADROUT:
7051   7B                        LD      A,E             ; partea mai putin semnificativa a adresei
7052   D3 A0                     OUT     (LEDPORT),A     ; se trimite la portul LEDOUT
7054   3E B7                     LD      A,RAMEMES       ; iar bitii A8 si A9 la portul SYSOUT
7056   E6 FC                     AND     11111100B       ; ( pe bitii 0 si 1 )
7058   B2                        OR      D
7059   D3 90                     OUT     (SYSOUT),A
705B                    ERRCYCLE:
705B   1A                        LD      A,(DE)          ; si se intra intr-un ciclu infinit de citire
705C   12                        LD      (DE),A          ; si scriere a locatiei eronate
705D   18 FC                     JR      ERRCYCLE
                        ;
                        ;       Programare Z80-CTC
                        ;
705F                    CTCPROG:
705F   3E 97                     LD      A,CTCCMD        ; se programeaza circuitul Z80-CTC pentru
7061   D3 C0                     OUT     (CTC),A         ; intreruperi din 2 in 2 milisecunde
7063   3E FA                     LD      A,CTCCOUNT
7065   D3 C0                     OUT     (CTC),A
7067   3E F8                     LD      A,LOW INTTAB    ; se trimite la circuit partea mai putin
7069   D3 C0                     OUT     (CTC),A         ; semnificativa a adresei tabelei de
                                                         ; intreruperi
                        ;
                        ;       Programarea sistemului de intreruperi a microprocesorului
                        ;
706B                    INTMOD:
706B   ED 5E                     IM      2               ; se programeaza sistemul de intreruperi a
706D   3E 6F                     LD      A,HIGH INTTAB   ; microprocesorului ( modul 2,in I partea mai
706F   ED 47                     LD      I,A             ; semnificativa a adresei tabelei de
                                                         ; intreruperi
7071   21 78DC                   LD      HL,INT          ;
7074   22 6FF8                   LD      (INTTAB),HL

                        Initializarea variabilelor de lucru
```

```
                        ;
7077                    INTVAR:
7077    D9                      EXX                     ; se initializeaza variabilele utilizate de ru
7078    21 6E31                 LD      HL,LEDBUF       ; tina INT: in HL adresa curenta din LEDBUF,
707B    0E A0                   LD      C,LEDPORT       ; in C adresa portului de afisaj,
707D    06 08                   LD      B,BUFLEN        ; in B numaratorul locatiilor si se reseteaza
707F    1E 00                   LD      E,0             ; indicatorul de inmultire
7081    D9                      EXX
                        ;
                        ;       Initializarea porturilor SYSOUT si PRTBUF
                        ;
7082                    INITSOUT:
7082    3E BF                   LD      A,NORMSOUT      ; se initializeaza portul SYSOUT si copia lui
7084    32 6F27                 LD      (WITNESS),A     ; din memorie
7087    D3 90                   OUT     (SYSOUT),A
7089                    HEADPOZ:
7089    CD 78B1                 CALL    MOTON           ; se pozitioneaza capul de imprimare
708C    CD 78B6                 CALL    MOTOFF
                        ;
                        ;       Determinarea tipului de pornire
                        ;
708F                    STRTTYPE:
708F    21 78F9                 LD      HL,MESSAGE      ; se initializeaza registrii HL,DE si BC pentru
7092    11 6E39                 LD      DE,EDBUF        ; a compara o zona de EPROM cu o zona de RAM
7095    01 0009                 LD      BC,CONTRLEN     ; pentru a determina daca este pornire rece sau
7098                    CONTR:                          ; pornire calda
7098    1A                      LD      A,(DE)          ; se compara octetul curent din EPROM cu cel
7099    ED A1                   CPI                     ; din RAM
709B    13                      INC     DE
709C    E2 7118                 JP      PO,WARMSTRT     ; daca cele doua zone au fost gasite identice,
                                                        ; salt la pornire calda
709F    28 F7                   JR      Z,CONTR         ; daca octetii verificati deja au fost egali,
                                                        ; dar mai sint octeti de comparat,salt
70A1                    COLDSTRT:
70A1    31 6FF0                 LD      SP,STACK2       ; se initializeaza stiva
                        ;
                        ;       Test de RAM nr. 2
                        ;
70A4                    RAMT2:
70A4    21 6C00                 LD      HL,RAMBOT       ; in HL adresa de inceput a memoriei RAM
```

```
70A7  01 0400          LD      BC,RAMLEN       ; in BC numarul de octeti
70AA                   RAMWR:
70AA  75               LD      (HL),L          ; se inscrie in octetul curent adresa ei
70AB  23               INC     HL              ; ( partea mai putin semnificativa )
70AC  0B               DEC     BC
70AD  74               LD      (HL),H          ; se inscrie in octetul curent adresa ei
70AE  23               INC     HL              ; ( partea mai semnificativa )
70AF  0B               DEC     BC
70B0  78               LD      A,B             ; se testeaza daca s-a terminat scrierea
70B1  B1               OR      C
70B2  20 F6            JR      NZ,RAMWR        ; daca nu,salt inapoi
                                               ; urmeaza recitirea valorilor inscrise
70B4  21 6C00          LD      HL,RAMBOT       ; in HL adresa de inceput a memoriei RAM
70B7  01 0400          LD      BC,RAMLEN       ; in BC numarul de octeti
70BA                   RAMRD:
70BA  7E               LD      A,(HL)          ; se citeste octetul curent si
70BB  BD               CP      L               ; se testeaza daca este corect
70BC  C2 7031          JP      NZ,RAMERR       ; daca nu,salt la eroare RAM
70BF  23               INC     HL
70C0  0B               DEC     BC
70C1  7E               LD      A,(HL)          ; se citeste octetul curent si
70C2  BC               CP      H               ; se testeaza daca este corect
70C3  C2 7031          JP      NZ,RAMERR       ; daca nu,salt la eroare RAM
70C6  23               INC     HL
70C7  0B               DEC     BC
70C8  78               LD      A,B             ; se testeaza daca s-a terminat citirea
70C9  B1               OR      C
70CA  20 EE            JR      NZ,RAMRD        ; daca nu,salt inapoi
                       ;
                       ;       Test de afisaj
                       ;
70CC                   LAMPT:
70CC  06 07            LD      B,7             ; numaratorul LED-urilor
70CE  AF               XOR     A               ; se trimite comanda de aprindere a tuturor
70CF  D3 A0            OUT     (LEDPORT),A     ; segmentelor la portul de afisaj
70D1                   LMP:
70D1  3A 6F27          LD      A,(WITNESS)
70D4  E6 F8            AND     MASK2
70D6  B0               OR      B
70D7  32 6F27          LD      (WITNESS),A
```

```
        MACRO-80 5.80   15-Jun-86       PAGE    1-8

70DA    D3 90                           OUT     (SYSOUT),A
70DC    11 3A98                         LD      DE,15000
70DF    CD 7817                         CALL    DELAYO
70E2    10 ED                           DJNZ    LMP
                        ;
                        ;       Initializarea zonelor din RAM
                        ;
70E4                    RAMINIT:
70E4    21 6C00                         LD      HL,RAMBOT       ; sumele si buferul EBCD se initializeaza
70E7    11 6C01                         LD      DE,RAMBOT+1     ; cu 0
70EA    36 00                           LD      (HL),0
70EC    01 0217                         LD      BC,BCDLEN*105+EBCDLEN
70EF    ED B0                           LDIR
70F1    21 6E17                         LD      HL,PRTBUF       ; se introduc spatii in buferul PRTBUF
70F4    06 12                           LD      B,PRTLEN
70F6    0E 20                           LD      C,BLANK
70F8    CD 774A                         CALL    FILLBYTS
70FB    CD 771F                         CALL    CLBUFDP         ; spatii in buferele KEYBUF si LEDBUF
70FE    CD 774F                         CALL    EXPAND          ; textele din EPROM se transfera in RAM,
                                                                ; rindurile completate cu spatii
7101    21 6EB3                         LD      HL,TICKNR       ; numaratorul bonurilor se init. cu 0
7104    06 04                           LD      B,TCKNRLEN
7106    0E 00                           LD      C,0
7108    CD 774A                         CALL    FILLBYTS
710B    21 78DC                         LD      HL,INT          ; se reface tabela INTTAB
710E    22 6FF8                         LD      (INTTAB),HL
7111    AF                              XOR     A
7112    CD 76B3                         CALL    DISPNUM         ; se afiseaza 0 si
7115    FB                              EI                      ; se valideaza sistemul de intreruperi
7116    18 14                           JR      MAINLOOP        ; salt la bucla principala a programului
                        ;
                        ;       Pornire calda
                        ;
7118                    WARMSTRT:
7118    ED 7B 6F28                      LD      SP,(STACKR)     ; se reface din RAM continutul registrului
                                                                ; indicator de virf de stiva
711C    E1                              POP     HL              ; se reface continutul registrilor
711D    D1                              POP     DE
711E    C1                              POP     BC
711F    F1                              POP     AF
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-9

7120   D9                       EXX
7121   08                       EX      AF,AF
7122   FD E1                    POP     IY
7124   DD E1                    POP     IX
7126   E1                       POP     HL
7127   D1                       POP     DE
7128   C1                       POP     BC
7129   F1                       POP     AF
712A   FB                       EI
712B   C9                       RET                     ; se valideaza sistemul de intreruperi
                                                        ; si se revine la adresa la care s-a intrerupt
                                                        ; executia programului
                        ;       Bucla principala a programului
                        ;
712C                    MAINLOOP:
712C   CD 7771                   CALL    INKEY           ; se citeste o tasta de la tastatura si
712F   FE 0B                    CP      NRORPT          ; si se testeaza daca este cifra sau punct
7131   38 04                    JR      C,SIMPBUY       ; daca da,salt la lansarea secventei de emitere
                                                        ; a unui bon client
7133   FE 0E                    CP      TOTAL           ; daca nu,se testeaza daca este TOTAL
7135   20 07                    JR      NZ,TESTFUNC     ; daca nu,urmeaza sa testam daca este FUNC
7137                    SIMPBUY:                        ; daca da,incepe emiterea unui bon client
7137   0E 00                    LD      C,0             ; C = 0,pentru a indica rutinei BUYTICK ca
7139   CD 7145                  CALL    BUYTICK         ; primul pret este fara cod
713C   18 EE                    JR      MAINLOOP        ; dupa terminarea bonului,se revine la inceput
713E                    TESTFUNC:
713E   FE 0B                    CP      FUNC            ; se testeaza daca s-a actionat FUNC
7140   CC 715B                  CALL    Z,PROCFUNC      ; daca da,incepe prelucrarea ei
7143   18 E7                    JR      MAINLOOP        ; dupa terminarea operatiei incepute se revine
                                                        ; la inceput
                                PAGE
```

```
MACR0-80 5.80   15-Jun-86      PAGE  1-10

                        ;
                        ;
                        ;       Rutina BUYTICK
                        ;
                        ; Executa toate operatiile necesare pentru eliberarea unui bon client:
                        ; citeste preturile introduse de la tastatura si le prelucreaza,iar la
                        ; sfirsit,la apasarea tastei TOTAL emite bonul creat.
                        ; Poate fi apelat din programul principal sau din rutina PROCFUNC.
                        ; Primeste de la rutina apelanta codul ultimei taste actionate in A si
                        ; numarul de apasari ale tastei FUNC in C.
7145                    BUYTICK:
7145    CD 7735                 CALL    CLDISP          ; sterge buferul de afisaj LEDBUF
7148                    BTESTTOT:
7148    FE OE                   CP      TOTAL           ; testeaza daca ultima tasta a fost TOTAL
714A    28 0B                   JR      Z,ENDBUYT       ; daca da,atunci salt la terminarea bonului
714C    CD 72E3                 CALL    INPRICE         ; daca nu,se citeste pretul inceput de la
714F    CD 7318                 CALL    PRPRICE         ; tastatura si se prelucreaza,dupa care
7152    CD 7358                 CALL    NXPRICE         ; se incepe citirea urmatorului pret sau a
7155    18 F1                   JR      BTESTTOT        ; tastei TOTAL si salt inapoi la test
7157                    ENDBUYT:                        ; daca s-a apasat TOTAL,atunci se termina si
7157    CD 7458                 CALL    EMITBUYT        ; se emite bonul creat
715A    C9                      RET
                                PAGE
```

```
                        ;
                        ;       Rutina PROCFUNC
                        ;
                        ; Este apelata din programul principal in cazul actionarii tastei FUNC.
                        ; Numara actionarile tastei FUNC in registrul C.Aceasta numarare se termina
                        ; la actionarea unei taste diferite de FUNC ( exceptie CLEAR ) si
                        ; se transfera controlul rutinei selectate prin valoarea din C.
                        ;
715B                    PROCFUNC:
715B  CD 7735                   CALL    CLDISP          ; se sterge buferul de afisaj
715E  0E 00                     LD      C,0             ; se initializeaza numaratorul actionarilor
                                                        ; tastei FUNC
7160  06 07                     LD      B,FUNCNUM       ; si numarul maxim admis de actionari
7162                    PROCLOOP:
7162  CD 7194                   CALL    CNTFUNC         ; se prelucreaza tasta FUNC
7165  CD 7771                   CALL    INKEY           ; si se citeste o noua tasta
7168  FE 0B                     CP      FUNC            ; se testeaza daca este FUNC
716A  28 F6                     JR      Z,PROCLOOP      ; daca da,se revine pentru prelucrarea ei
716C  CD 7735                   CALL    CLDISP          ; daca nu,se sterge afisajul si
716F  FE 0F                     CP      CLEAR           ; se testeaza daca este CLEAR
7171  20 05                     JR      NZ,CASE         ; daca nu,salt la ramificare
7173                    PROCCL:                         ; daca a fost CLEAR,
7173  AF                        XOR     A               ; atunci se afiseaza 0 si
7174  CD 76B3                   CALL    DISPNUM         ; se revine in programul principal
7177  C9                        RET
7178                    CASE:
7178  21 7186                   LD      HL,CASETAB      ; in HL adresa tabelului cu adresele rutinelor
717B  0D                        DEC     C               ; se decrementeaza numaratorul pentru ca valo-
717C  06 00                     LD      B,0             ; rile sa inceapa de la 0 ( pe 16 biti in BC )
717E  09                        ADD     HL,BC           ; se determina adresa la care se gaseste
717F  09                        ADD     HL,BC           ; adresa rutinei de apelat
7180  5E                        LD      E,(HL)          ; se transfera in E si D adresa rutinei,
7181  23                        INC     HL
7182  56                        LD      D,(HL)
7183  EB                        EX      DE,HL           ; iar din DE in HL
7184  0C                        INC     C               ; se reface valoarea initiala a numaratorului
7185  E9                        JP      (HL)            ; si salt la rutina selectata
```

```
MACRO-80 5.80   15-Jun-86      PAGE    1-12

7186                        CASETAB:            ; tabelul cu adresele rutinelor
7186    7145                        DW      BUYTICK
7188    7145                        DW      BUYTICK
718A    71B5                        DW      SETUP
718C    7219                        DW      ERATICK
718E    7244                        DW      TOTSORT
7190    7282                        DW      TOTDAY
7192    72BD                        DW      SYNTH
                                    PAGE
```

```
                        ;       Rutina CNTFUNC
                        ;
                        ; Prelucreaza o actionare a tastei FUNC ( incrementare numarator,afisare,
                        ; sonerie ).
                        ; Distruge: C,D,E,H,L
                        ; Registrii utilizati:
                        ;       C - numaratorul actionarilor tastei FUNC
                        ;       DE- pentru adresarea buferului de afisaj LEDBUF
                        ;       HL- pentru adresarea generatorului de cifre pe 7 segmente
                        ;       B - numarul maxim admis de actionari
                        ;
7194                    CNTFUNC:
7194  F5                        PUSH    AF              ; se salveaza A
7195  0C                        INC     C               ; se incrementeaza numaratorul actionarilor
                                                        ; tastei FUNC
7196  78                        LD      A,B             ; se transfera numarul maxim de actionari
                                                        ; admis in A
7197  B9                        CP      C               ; se testeaza daca numaratorul a depasit
                                                        ; aceasta valoare
7198  30 02                     JR      NC,DISPNR       ; daca nu,salt la afisarea numaratorului
719A  0E 01                     LD      C,1             ; daca da,se reinitializeaza numaratorul
719C                    DISPNR:
719C  CD 7735                   CALL    CLDISP          ; se sterge buferul de afisaj
719F  06 00                     LD      B,0
71A1  11 6E31                   LD      DE,LEDBUF+0     ; in DE adresa din buferul de afisaj unde se va
                                                        ; introduce codul valorii numaratorului
71A4  21 796C                   LD      HL,LEDGEN       ; in HL adresa generatorului de cifre pe 7
                                                        ; segmente
71A7  09                        ADD     HL,BC           ; se determina adresa codului corespunzator
                                                        ; valorii numaratorului
71A8  47                        LD      B,A             ; se reface in B valoarea maxima admisa
71A9  7E                        LD      A,(HL)          ; codul selectat se transfera
71AA  12                        LD      (DE),A          ; in buferul de afisaj
71AB  21 0BB8                   LD      HL,FCSDFRE      ; si se suna soneria
71AE  3E 14                     LD      A,FCSDTIME
71B0  CD 77C3                   CALL    BEEP
71B3  F1                        POP     AF              ; se reface A
71B4  C9                        RET
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-14

                        ;
                        ;
                        ;       Rutina SETUP
                        ;
                        ; Este rutina prin care se introduc in EDBUF informatii despre casa de marcat
                        ; ( numarul unitatii comerciale,numarul casei,data si numarul casierului ).
                        ;
71B5                    SETUP:
71B5    01 0000                 LD      BC,0            ; se initializeaza numaratorul de programare
71B8                    SETN:
71B8    FE 0B                   CP      NRORPT          ; se testeaza daca s-a introdus cifra sau punct
71BA    38 0A                   JR      C,PARAMPR       ; daca da,salt la prelucrare
71BC    CD 7771                 CALL    INKEY           ; daca nu,se citeste o noua tasta
71BF    FE 0F                   CP      CLEAR           ; se testeaza daca este CLEAR
71C1    CC 771F                 CALL    Z,CLBUFDP       ; daca da,se reinitializeaza buferele
71C4    18 F2                   JR      SETN
71C6                    PARAMPR:
71C6    CD 76D9                 CALL    STORDISP        ; cifra sau punctul se introduce in  buferele
                                                        ; KEYBUF si LEDBUF
71C9    CD 7771                 CALL    INKEY           ; se citeste o noua tasta
71CC    FE 0E                   CP      TOTAL           ; si se testeaza daca este TOTAL
71CE    20 E8                   JR      NZ,SETN         ; daca nu,continua introducerea parametrului
                                                        ; curent
71D0    CD 71EB                 CALL    TRANSPAR        ; daca este TOTAL,se transfera parametrul la
                                                        ; locul lui
71D3    CD 771F                 CALL    CLBUFDP         ; si se sterg buferele KEYBUF si LEDBUF
71D6    0C                      INC     C               ; se incrementeaza numaratorul de programare
71D7    79                      LD      A,C             ; si se testeaza daca a ajuns la 4
71D8    FE 04                   CP      4
71DA    3E 0E                   LD      A,TOTAL
71DC    20 DA                   JR      NZ,SETN         ; daca nu,incepe programarea urmatorului
                                                        ; parametru
71DE    AF                      XOR     A               ; daca da,programarea s-a terminat si se
71DF    CD 76B3                 CALL    DISPNUM         ; afiseaza 0
71E2    21 0320                 LD      HL,SENDFR       ; se suna soneria
71E5    3E 14                   LD      A,SENDTM
71E7    CD 77C3                 CALL    BEEP
71EA    C9                      RET
                        ;
                        ;
```

```
MACRO-80 5.80   15-Jun-86      PAGE   1-15

                        ;
                        ;       Rutina TRANSPAR
                        ;
                        ; Apelata din SETUP,transfera parametrii introdusi la adresele lor,adrese
                        ; determinate de valoarea numaratorului de programare.
                        ; Transferul incepe cu ultima cifra introdusa,deci tabela SETUPTAB contine
                        ; adresele de sfirsit ale zonelor.
                        ; Distruge: A,D,E,H,L ( B = 0 si la inceput,si la sfirsit )
                        ; Registrii utilizati:
                        ;       HL- pentru adresarea tabelei SETUPTAB si a zonei parametrului
                        ;       DE- la formarea adresei zonei parametrului si pentru adresarea
                        ;           buferului KEYBUF
                        ;       B - la calcularea adresei din SETUPTAB si ca numarator al octetilor
                        ;           parametrului
                        ;       A - la diferite operatii
                        ;
71EB                    TRANSPAR:
71EB    21 720D                 LD      HL,SETUPTAB     ; in HL adresa tabelei de adrese si lungimi
71EE    09                      ADD     HL,BC           ; se calculeaza adresa corespunzatoare
71EF    09                      ADD     HL,BC           ; valorii numaratorului
71F0    09                      ADD     HL,BC
71F1    5E                      LD      E,(HL)          ; se transfera adresa parametrului din
71F2    23                      INC     HL              ; tabela SETUPTAB in DE
71F3    56                      LD      D,(HL)
71F4    23                      INC     HL              ; iar lungimea in octeti a parametrului
71F5    46                      LD      B,(HL)          ; in B
71F6    EB                      EX      DE,HL           ; adresa din DE in HL
71F7    11 6E30                 LD      DE,KEYBUF+BUFLEN-1      ; in DE adresa buferului KEYBUF
71FA                    TAKEBYTE:
71FA    1A                      LD      A,(DE)          ; se ia octetul curent din KEYBUF
71FB    1B                      DEC     DE
71FC    CB 7F                   BIT     7,A             ; si se testeaza daca contine punct
71FE    28 08                   JR      Z,NOPOINT       ; daca nu,salt
7200    36 0A                   LD      (HL),POINT      ; daca da,se introduce punctul
7202    2B                      DEC     HL
7203    CB BF                   RES     7,A             ; se reseteaza bitul indicator de punct
7205    05                      DEC     B               ; se decrementeaza numaratorul de octeti
7206    28 04                   JR      Z,ENDTRAN       ; daca s-a ajuns la 0,salt la sfirsit
7208                    NOPOINT:                        ; daca nu,atunci
7208    77                      LD      (HL),A          ; se transfera cifra
7209    2B                      DEC     HL
```

```
MACRO-80 5.80   15-Jun-86        PAGE    1-16

720A   10 EE                     DJNZ    TAKEBYTE        ; se decrementeaza numaratorul si daca nu s-a
                                                         ; ajuns la 0,salt inapoi pentru un nou octet
                                                         ; din KEYBUF
720C                     ENDTRAN:
720C   C9                        RET
720D                     SETUPTAB:
720D   6EF2                      DW      SHOPN+SHOPNL-1  ; adresa numar unitate
720F   03                        DB      SHOPNL          ; lungime numar unitate
7210   6F02                      DW      DESKN+DESKNL-1  ; adresa numar casa
7212   02                        DB      DESKNL          ; lungime numar casa
7213   6F11                      DW      DATE+DATEL-1    ; adresa data
7215   08                        DB      DATEL           ; lungime data
7216   6EBF                      DW      CLRKN+CLRKNL-1  ; adresa numar casier
7218   03                        DB      CLRKNL          ; lungime numar casier
                                 PAGE
```

```
MACRO-80 5.80   15-Jun-86      PAGE    1-17


                        ;
                        ;
                        ;       Rutina ERATICK
                        ;
                        ;
7219                    ERATICK:
7219    0E 00                   LD      C,0             ; primul pret in caz de anulare trebuie sa fie
                                                        ; fara cod,pentru ca FUNC este utilizat pentru
                                                        ; a specifica anularea;deci C,care la urmatoa-
                                                        ; rele preturi va fi numaratorul actionarilor
721B    57                      LD      D,A             ; tastei FUNC,se initializeaza cu 0
721C    DB 90                   IN      A,(SYSIN)       ; se salveaza codul ultimei taste
721E    CB 4F                   BIT     KEY1,A          ; si se citeste portul intrarilor
7220    28 15                   JR      Z,ERAERR        ; si se testeaza prezenta cheii de control
7222                    ERACOR:                         ; daca lipseste,eroare
7222    7A                      LD      A,D             ; la operare corecta
7223                    ETESTTOT:                       ; se reface codul ultimei taste
7223    FE 0E                   CP      TOTAL           ; si se testeaza daca este TOTAL
7225    28 0B                   JR      Z,ENDERAT       ; daca da,salt la emiterea bonului de anulare
7227    CD 72E3                 CALL    INPRICE         ; daca nu,se citeste pretul inceput de la
722A    CD 732C                 CALL    ERPRICE         ; tastatura si se prelucreaza,dupa care
722D    CD 7358                 CALL    NXPRICE         ; se incepe citire urmatorului pret sau a
7230    18 F1                   JR      ETESTTOT        ; tastei TOTAL si salt inapoi la test
7232                    ENDERAT:                        ; daca s-a apasat TOTAL,atunci se termina si
7232    CD 749F                 CALL    EMITERAT        ; se emite bonul creat
7235    18 0C                   JR      ERAEND
7237                    ERAERR:
7237    21 03E8                 LD      HL,OPERFR       ; in caz de operare gresita,se suna soneria
723A    3E 14                   LD      A,OPERTM
723C    CD 77C3                 CALL    BEEP
723F    AF                      XOR     A               ; si se afiseaza 0
7240    CD 76B3                 CALL    DISPNUM
7243                    ERAEND:
7243    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-18


                        ;       Rutina TOTSORT
                        ;
                        ; Imprima si afiseaza totalul corespunzator sortimentului selectat.Nu se
                        ; executa numai in prezenta cheii de control.La apel in A codul ultimei taste
                        ; actionate,KEYBUF si LEDBUF sterse.Daca se opereaza incorect,se suna soneria
                        ; si se afiseaza 0.
7244                    TOTSORT:
7244    CD 7370                 CALL    INCODE          ; se citeste codul si inca o tasta
7247    FE 0E                   CP      TOTAL           ; este aceasta tasta TOTAL ?
7249    20 2A                   JR      NZ,ERRTSORT     ; daca nu,eroare
724B    DB 90                   IN      A,(SYSIN)       ; se citeste portul intrarilor
724D    CB 4F                   BIT     KEY1,A          ; se testeaza prezenta cheii de control
724F    28 24                   JR      Z,ERRTSORT      ; daca nu este prezenta,eroare
7251    CD 740F                 CALL    SORTADR         ; se formeaza in HL adresa totalului cautat,
7254    11 6E1B                 LD      DE,PRTBUF+4     ; totalul se despacheteaza si se introduce
7257    CD 7666                 CALL    BCDCI           ; in buferul PRTBUF,
725A    CD 76F3                 CALL    DISPSUM         ; de unde se afiseaza
725D    06 04                   LD      B,TCKNRLEN      ; se incrementeaza numarul de bon
725F    21 6EB6                 LD      HL,TICKNR+TCKNRLEN-1
7262    CD 75E7                 CALL    EBCDINC
7265    CD 740F                 CALL    SORTADR         ; totalul se introduce in SUM
7268    11 6E93                 LD      DE,SUM
726B    CD 7666                 CALL    BCDCI
726E    0E 04                   LD      C,SORTMASK      ; in C masca de selectie a liniilor din EDBUF
7270    CD 74E6                 CALL    PRTTICK         ; si se imprima bonul
7273    18 0C                   JR      ENDTSORT
7275                    ERRTSORT:
7275    21 03E8                 LD      HL,OPERFR       ; in caz de eroare,se suna soneria
7278    3E 14                   LD      A,OPERTM
727A    CD 77C3                 CALL    BEEP
727D    AF                      XOR     A               ; si se afiseaza 0
727E    CD 76B3                 CALL    DISPNUM
7281                    ENDTSORT:
7281    C9                      RET
                        PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE   1-19

                        ;       Rutina TOTDAY
                        ;
                        ; Imprima si afiseaza totalul zilei.Nu se executa decit in prezenta cheii de
                        ; control.La apel in A codul ultimei tasta actionate,iar KEYBUF si LEDBUF
                        ; sterse.In caz de operare gresita se suna soneria si 0 pe afisaj.
7282                    TOTDAY:
7282  FE 0E                     CP      TOTAL           ; dupa ultimul FUNC s-a actionat TOTAL ?
7284  20 2A                     JR      NZ,ERRTDAY      ; daca nu,eroare
7286  DB 90                     IN      A,(SYSIN)       ; se citeste portul de intrari
7288  CB 4F                     BIT     KEY1,A          ; si se testeaza prezenta cheii de control
728A  28 24                     JR      Z,ERRTDAY       ; daca nu este prezenta,eroare
728C  21 6C00                   LD      HL,DAYBCD       ; daca da,se despacheteaza totalul zilei si se
728F  11 6E1B                   LD      DE,PRTBUF+4     ; introduce in buferul de tiparire PRTBUF,
7292  CD 7666                   CALL    BCDCI
7295  CD 76F3                   CALL    DISPSUM         ; de unde se afiseaza
7298  06 04                     LD      B,TCKNRLEN      ; se incrementeaza numarul bonului
729A  21 6EB6                   LD      HL,TICKNR+TCKNRLEN-1
729D  CD 75E7                   CALL    EBCDINC
72A0  21 6C00                   LD      HL,DAYBCD       ; se introduce totalul in EDBUF
72A3  11 6E93                   LD      DE,SUM
72A6  CD 7666                   CALL    BCDCI
72A9  0E 02                     LD      C,DAYMASK       ; in C masca de selectie
72AB  CD 74E6                   CALL    PRTTICK         ; se tipareste bonul
72AE  18 0C                     JR      ENDTDAY
72B0                    ERRTDAY:
72B0  21 03E8                   LD      HL,OPERFR       ; in caz de eroare de operare,se suna soneria
72B3  3E 14                     LD      A,OPERTM
72B5  CD 77C3                   CALL    BEEP
72B8  AF                        XOR     A               ; si se afiseaza 0
72B9  CD 76B3                   CALL    DISPNUM
72BC                    ENDTDAY:
72BC  C9                        RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86   PAGE    1-20

                        ;       Rutina SYNTH
                        ;
                        ; Imprima toate totalurile de sortiment.Nu se executa numai in prezenta cheii
                        ; de control.In momentul apelului in A codul ultimei taste actionate,iar
                        ; KEYBUF si LEDBUF sterse.
72BD                    SYNTH:
72BD    FE 0E                   CP      TOTAL           ; dupa ultimul FUNC s-a actionat TOTAL ?
72BF    20 15                   JR      NZ,ERRSYNTH     ; daca nu,eroare
72C1    DB 90                   IN      A,(SYSIN)       ; se citeste portul de intrari
72C3    CB 4F                   BIT     KEY1,A          ; se testeaza prezenta cheii de control
72C5    28 0F                   JR      Z,ERRSYNTH      ; daca nu este prezenta,eroare
72C7    06 04                   LD      B,TCKNRLEN      ; se incrementeaza numarul bonului
72C9    21 6EB6                 LD      HL,TICKNR+TCKNRLEN-1
72CC    CD 75E7                 CALL    EBCDINC
72CF    0E 01                   LD      C,SYNTMASK      ; in C masca de selectie a liniilor
72D1    CD 74E6                 CALL    PRTTICK         ; se imprima bonul
72D4    18 08                   JR      ENDSYNTH
72D6                    ERRSYNTH:
72D6    21 03E8                 LD      HL,OPERFR       ; in caz de eroare,se suna soneria
72D9    3E 14                   LD      A,OPERTM
72DB    CD 77C3                 CALL    BEEP
72DE                    ENDSYNTH:
72DE    AF                      XOR     A               ; la sfirsit 0 pe afisaj
72DF    CD 76B3                 CALL    DISPNUM
72E2    C9                      RET
                                PAGE
```

```
                        ;
                        ;
                        ;       Rutina INPRICE
                        ;
                        ; Citeste un pret ( cu sau fara cod ) de la tastatura.Primeste de la rutina
                        ; apelanta codul ultimei taste actionate in A si numarul de actionari ale
                        ; tastei FUNC in C.
                        ; Introducerea pretului se termina la actionarea tastei "+".
                        ; La terminarea rutinei pretul introdus se gaseste in buferele
                        ; KEYBUF si LEDBUF,iar codul ( 99 in cazul preturilor fara cod explicit )
                        ; in buferul CODE.
72E3                    INPRICE:
72E3    57                      LD      D,A             ; se salveaza codul ultimei tasta actionate
72E4    79                      LD      A,C             ; si se testeaza daca inaintea ei s-a apasat
72E5    B7                      OR      A               ; FUNC ( se pozitioneaza indicatorul Z )
72E6    7A                      LD      A,D             ; se reface codul ultimei taste ( operatie ce
                                                        ; nu afecteaza indicatorul Z )
72E7    28 0C                   JR      Z,CODELESS      ; daca FUNC nu a fost actionat,atunci salt la
72E9                    CODED:                          ; pret fara cod ( implicit 99 )
72E9    CD 7370                 CALL    INCODE          ; daca FUNC a fost actionat,atunci este pret cu
72EC                    AFTERCOD:                       ; cod,deci se citeste codul si inca o tasta
72EC    FE 0B                   CP      NRORPT          ; dupa ea si se testeaza daca aceasta tasta
72EE    38 0F                   JR      C,PROCDIG       ; este numar sau punct
72F0    CD 7771                 CALL    INKEY           ; daca da,salt la prelucrarea ei
72F3    18 F7                   JR      AFTERCOD        ; in caz contrar se citeste o noua tasta si
72F5                    CODELESS:                       ; salt inapoi la testare
72F5    0E 09                   LD      C,9             ; in cazul pretului fara cod,se introduce
72F7    CD 739B                 CALL    IMPLCODE        ; codul 99 in zona CODE,iar variabila
72FA    21 6F2A                 LD      HL,NRFUNC       ; NRFUNC din RAM se initializeaza cu 1,pentru
72FD    36 01                   LD      (HL),1          ; a indica pret de produs rutinei OPFORBUY
72FF                    PROCDIG:
72FF    CD 76D9                 CALL    STORDISP        ; cifra sau punctul curent se memoreaza si
7302                    NEXTKEY:                        ; se afiseaza
7302    CD 7771                 CALL    INKEY           ; se citeste o noua tasta
7305    FE 0F                   CP      CLEAR           ; se testeaza daca este CLEAR
7307    CC 771F                 CALL    Z,CLBUFDP       ; daca da,se sterg buferele KEYBUF si LEDBUF
730A    FE 0D                   CP      ASTER           ; se testeaza daca este inmultire;daca da,
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-22

730C   CC 75CF                          CALL    Z,MULTOP        ; se executa operatiile de pregatire a
730F                            TENDP:                           ; inmultirii
730F   FE 0B                            CP      NRORPT          ; se testeaza daca este cifra sau punct
7311   38 EC                            JR      C,PROCDIG       ; daca da,salt la prelucrarea ei
7313   FE 0C                            CP      PLUS            ; se testeaza daca este PLUS
7315   20 EB                            JR      NZ,NEXTKEY      ; daca nu,se revine pentru o noua citire
7317   C9                               RET                     ; daca da,atunci introducerea pretului s-a
                                                                 ; terminat si se revine in rutina apelanta
                                ;
                                ;
                                ;       Rutina PRPRICE
                                ;
                                ; Prelucreaza pretul din buferul KEYBUF: il analizeaza si daca este corect,
                                ; atunci il impacheteaza,il aduna la totalul de sortiment corespunzator si
                                ; executa operatiile necesare pentru emiterea bonului final.
                                ;
7318                            PRPRICE:
7318   CD 75F2                          CALL    SYNTAN          ; se analizeaza pretul din buferul KEYBUF
731B   20 0E                            JR      NZ,ENDPRPR      ; daca il gaseste incorect,nu se mai executa
                                                                 ; nici o prelucrare,ci salt la sfirsit
731D   CD 7643                          CALL    PRELPROC        ; daca pretul este corect,il impacheteaza si
                                                                 ; daca este indicat,executa inmultirea
7320   CD 743A                          CALL    ADDSORT         ; aduna pretul la totalul de sortiment
                                                                 ; corespunzator codului introdus
7323   21 6E26                          LD      HL,PRTBUF+15    ; introduce semnul "+" in buferul PRTBUF
7326   36 0C                            LD      (HL),PLUS
7328   CD 73A8                          CALL    OPFORBUY        ; executa operatiile necesare bonului in curs
732B                            ENDPRPR:
732B   C9                               RET
                                        PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-23


                        ;
                        ;
                        ;       Rutina ERPRICE
                        ;
                        ; Prelucreaza pretul din KEYBUF pentru anulare: il analizeaza si daca este
                        ; corect,atunci il impacheteaza,il scade din totalul de sortiment corespunzator
                        ; si executa operatiile necesare pentru bonul de anulare.
                        ;
732C                    ERPRICE:
732C    CD 75F2                 CALL    SYNTAN          ; se analizeaza pretul din buferul KEYBUF
732F    20 26                   JR      NZ,ENDERPR      ; daca il gaseste incorect,nu se mai executa
                                                        ; nici o prelucrare,ci salt la eroare
7331    CD 7643                 CALL    PRELPROC        ; daca pretul este corect,il impacheteaza si
                                                        ; daca este indicat,executa inmultirea
7334    21 6E26                 LD      HL,PRTBUF+15    ; se introduce "A"
7337    36 10                   LD      (HL),ANULARE
7339    CD 7449                 CALL    SUBSORT         ; se scade pretul din totalul de sortiment
                                                        ; corespunzator codului din CODE
733C                    TESTSUB:                        ; se testeaza corectitudinea rezultatului
733C    38 05                   JR      C,SUBERR        ; daca eronat,salt la eroare
733E                    SUBCOR:                         ; daca rezultatul este corect,
733E    CD 73A8                 CALL    OPFORBUY        ; se executa operatiile necesare emiterii
7341    18 14                   JR      ENDERPR         ; bonului
7343                    SUBERR:                         ; daca s-a detectat o eroare,
7343    21 6DF9                 LD      HL,GUESTBCD     ; vechiul total ( de anulare ) al clientului
7346    11 6E1B                 LD      DE,PRTBUF+4     ; se despacheteaza si se transfera in PRTBUF
7349    CD 7666                 CALL    BCDCI           ; de unde
734C    CD 76F3                 CALL    DISPSUM         ; se afiseaza
734F    21 07D0                 LD      HL,ERSDFR       ; si se suna soneria
7352    3E 14                   LD      A,ERSDTM
7354    CD 77C3                 CALL    BEEP
7357                    ENDERPR:
7357    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-24

                        ;
                        ;
                        ;       Rutina NXPRICE
                        ;
                        ; Citeste inceputul unui pret sau tasta TOTAL.
                        ; Registrii utilizati:
                        ;       C - numaratorul actionarilor tastei FUNC
                        ;       B - numarul maxim de actionari admis
                        ;
7358                    NXPRICE:
7358    0E 00                   LD      C,0             ; se initializeaza numaratorul actionarilor
                                                        ; tastei FUNC si
735A    06 02                   LD      B,MAXFUNC2      ; se fixeaza numarul maxim de actionari
735C                    BEGPR:
735C    CD 7771                 CALL    INKEY           ; se citeste o tasta si
735F    FE 0B                   CP      FUNC            ; se testeaza daca este FUNC
7361    CC 7194                 CALL    Z,CNTFUNC       ; daca da,se contorizeaza
7364    FE 0B                   CP      NRORPT          ; se testeaza daca este cifra sau punct
7366    38 04                   JR      C,ENDNXPR       ; daca da,salt la sfirsit
7368    FE 0E                   CP      TOTAL           ; se testeaza daca este TOTAL
736A    20 F0                   JR      NZ,BEGPR        ; daca nu,se revine pentru o noua citire
736C                    ENDNXPR:
736C    CD 771F                 CALL    CLBUFDP         ; se sterg KEYBUF si LEDBUF
736F    C9                      RET
                                PAGE
```

```
                        ;       Rutina INCODE
                        ;
                        ; Citeste de la tastatura un cod de produs sau de ambalaj.Primeste de la
                        ; rutina apelanta in A codul tastei apasate dupa FUNC si in C numarul de
                        ; actionari ale tastei FUNC.
                        ; Transmite rutinei apelante prin A codul tastei apasate dupa ultima cifra
                        ; a codului.Distruge: A,B,C,D,E,H,L
                        ; Registrii utilizati:
                        ;       A - pentru comunicatii intre rutine
                        ;       B - numarator al cifrelor codului la introducerea lor si la
                        ;           transferul lor din KEYBUF in CODE ( aici impreuna cu C )
                        ;       DE- pentru adresarea zonei CODE la transfer
                        ;       HL- pentru adresarea variabilei NRFUNC si a buferului KEYBUF
7370                    INCODE:
7370    21 6F2A                 LD      HL,NRFUNC       ; numarul de actionari ale tastei FUNC se
7373    71                      LD      (HL),C          ; memoreaza in variabila NRFUNC din RAM,
                                                        ; pentru rutina OPFORBUY.
7374    06 02                   LD      B,SCLEN         ; in B numarul de cifre ale codului
7376                    CODDIG:
7376    FE 0A                   CP      POINT           ; se testeaza daca ultima tasta actionata a
                                                        ; fost cifra
7378    38 05                   JR      C,PRCODDIG      ; daca da,salt la prelucrarea ei
737A    CD 7771                 CALL    INKEY           ; daca nu,se citeste o noua tasta si
737D    18 F7                   JR      CODDIG          ; se revine pentru testarea ei
737F                    PRCODDIG:
737F    CD 76D9                 CALL    STORDISP        ; cifra curenta se memoreaza si se afiseaza si
7382    CD 7771                 CALL    INKEY           ; se citeste o noua tasta
7385    FE 0F                   CP      CLEAR           ; se testeaza daca este CLEAR si daca da,atunci
7387    CC 7744                 CALL    Z,CLEARCOD      ; se sterg buferele de tastatura si de afisaj
                                                        ; si se reinitializeaza numaratorul de cod
738A    10 EA                   DJNZ    CODDIG          ; se decrementeaza numaratorul cifrelor codului
                                                        ; si daca codul nu este complet,salt inapoi
738C    21 6E2F                 LD      HL,KEYBUF+6     ; codul introdus se transfera din buferul de
738F    11 6E57                 LD      DE,CODE         ; tastatura KEYBUF in zona CODE si
7392    01 0002                 LD      BC,SCLEN
7395    ED B0                   LDIR
7397    CD 771F                 CALL    CLBUFDP         ; se sterg buferele KEYBUF si LEDBUF
739A    C9                      RET
```

```
MACRO-80 5.80   15-Jun-86        PAGE    1-26

                                ;
                                ;
                                ;       Rutina IMPLCODE
                                ;
                                ; Initializeaza octetii zonei CODE cu cifra primita in registrul C.
                                ; Nu distruge nici un registru.
                                ; Registrii utilizati:
                                ;       B - numarator ai octetilor zonei
                                ;       HL- pentru adresarea zonei
                                ;       C - contine cifra cu care se initializeaza zona
 739B                           IMPLCODE:
 739B   C5                              PUSH    BC              ; se salveeaza registrii utilizati
 739C   E5                              PUSH    HL
 739D   06 02                           LD      B,SCLEN         ; in B lungimea zonei CODE
 739F   21 6E57                         LD      HL,CODE         ; in HL adresa zonei si
 73A2   CD 774A                         CALL    FILLBYTS        ; se apeleaza rutina de initializare
 73A5   E1                              POP     HL
 73A6   C1                              POP     BC              ; se refac registrii salvati
 73A7   C9                              RET
                                        PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-27


                                ;
                                ;
                                ;       Rutina OPFORBUY
                                ;
                                ; Executa operatiile legate de bonul clientului.
                                ; Testeaza valoarea variabilei NRFUNC din RAM: daca NRFUNC=1,pretul este pret
                                ; de marfa,deci se aduna la totalul clientului,daca NRFUNC=2,pretul este pret
                                ; de ambalaj,deci se scade din totalul clientului.Daca operatia a fost corect
                                ; executata,se afiseaza noua suma a clientului si se tipareste pretul pe bon.
                                ; Daca nu,atunci se scade pretul din totalul de sortiment corespunzator
                                ; ( pentru a anula adunarea care s-a efectuat deja cu ADDSORT ) si se afiseaza
                                ; vechea suma a clientului.
                                ;
73A8                            OPFORBUY:
73A8    CD 7735                         CALL    CLDISP
73AB    21 6DF9                         LD      HL,GUESTBCD     ; se transfera suma clientului in TMPBCD
73AE    CD 7705                         CALL    MOVTMP
73B1    3A 6F2A                         LD      A,(NRFUNC)      ; se incarca NRFUNC in A si
73B4    3D                              DEC     A               ; se decrementeaza
73B5    20 0C                           JR      NZ,PACKAGE      ; daca nu este 0 ( deci a fost 2 ),salt la
                                                                ; prelucrare pret de ambalaj
73B7                            PRODUCT:                        ; daca este pret de marfa,atunci
73B7    3A 6E57                         LD      A,(CODE)        ; se testeaza incadrarea codului introdus
73BA    B7                              OR      A               ; intre limitele 10 - 99
73BB    37                              SCF                     ; se pozitioneaza un indicator posibil de er.
73BC    28 14                           JR      Z,TESTER        ; la neincadrare,salt la testarea erorii
73BE    CD 755D                         CALL.   ADDBCD          ; se aduna la totalul clientului,in TMPBCD si
73C1    18 0F                           JR      TESTER          ; salt la testarea corectitudinii
73C3                            PACKAGE:                        ; in caz de pret de ambalaj
73C3    3A 6E57                         LD      A,(CODE)        ; se testeaza incadrarea codului introdus
73C6    B7                              OR      A               ; intre limitele 0 - 9
73C7    37                              SCF                     ; se pozitioneaza un indicator posibil de er.
73C8    20 08                           JR      NZ,TESTER       ; la neincadrare,salt la testarea erorii
73CA    21 6E26                         LD      HL,PRTBUF+15    ; se schimba semnul "+" in "-" si
73CD    36 22                           LD      (HL),MINUS
73CF    CD 7575                         CALL    SUBBCD          ; se scade pretul din totalul clientului
73D2                            TESTER:                         ; ( CY va indica daca rezultat corect sau nu )
73D2    38 2B                           JR      C,OPERROR       ; daca rezultat eronat,salt
```

```
MACRO-80 5.80   15-Jun-86   PAGE   1-28

73D4                    CORRECT:
73D4  11 6DF9                   LD      DE,GUESTBCD     ; rezultatul adunarii/scaderii se transfera
73D7  CD 7712                   CALL    TMPMOV          ; in buferul totalului clientului
73DA  21 6DF9                   LD      HL,GUESTBCD     ; totalul clientului se despacheteaza si
73DD  11 6E1B                   LD      DE,PRTBUF+4     ; se transfera in buferul de imprimare
73E0  CD 7666                   CALL    BCDCI           ; PRTBUF,de unde
73E3  CD 76F3                   CALL    DISPSUM         ; se afiseaza
73E6  21 6E08                   LD      HL,DATBCD       ; pretul introdus se despacheteaza si
73E9  11 6E1B                   LD      DE,PRTBUF+4     ; se transfera in buferul de imprimare
73EC  CD 7666                   CALL    BCDCI           ; PRTBUF
73EF  21 6E57                   LD      HL,CODE         ; codul de produs din CODE se transfera
73F2  11 6E18                   LD      DE,PRTBUF+1     ; tot in PRTBUF,inaintea pretului
73F5  01 0002                   LD      BC,SCLEN
73F8  ED B0                     LDIR
73FA  CD 783A                   CALL    PRTLINE         ; se imprima linia formata
73FD  18 0F                     JR      ENDOP
73FF                    OPERROR:                        ; daca a aparut o eroare,
73FF  CD 7449                   CALL    SUBSORT         ; se scade pretul din totalul de sortiment
7402  21 6DF9                   LD      HL,GUESTBCD     ; si se transfera vechiul total al clientului
7405  11 6E1B                   LD      DE,PRTBUF+4     ; in buferul de imprimare PRTBUF,
7408  CD 7666                   CALL    BCDCI
740B  CD 76F3                   CALL    DISPSUM         ; de unde se afiseaza
740E                    ENDOP:
740E  C9                        RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-29


                        ;
                        ;
                        ;       Rutina SORTADR
                        ;
                        ; Formeaza in HL adresa totalului de sortiment corespunzator codului din CODE.
                        ; Distruge: H,L
                        ; Registrii utilizati:
                        ;       IY- pentru adresarea zonei CODE
                        ;       B - transmite numarul de adunari de efectuat rutinei ADDD
                        ;       DE- cifrele codului pe 16 biti
                        ;       HL- la efectuarea adunarilor
                        ;
740F                    SORTADR:
740F    C5                      PUSH    BC              ; se salveaza registrii utilizati
7410    D5                      PUSH    DE
7411    FD E5                   PUSH    IY
7413    FD 21 6E57              LD      IY,CODE         ; in IY adresa zonei CODE
7417    FD 5E 00                LD      E,(IY+0)        ; se transfera cifra mai semnificativa a codu-
741A    16 00                   LD      D,0             ; lui in E ( deci in DE pe 16 biti ) si se
741C    21 0000                 LD      HL,0            ; aduna la HL de 10 ori,deci la sfirsitul
741F    06 0A                   LD      B,10            ; secventei in HL avem cifra x 10
7421    CD 7436                 CALL    ADDD
7424    FD 5E 01                LD      E,(IY+1)        ; se transfera in E cifra mai putin semni-
                                                        ; ficativa a codului ( in DE pe 16 biti )
7427    19                      ADD     HL,DE           ; si se aduna la HL,deci in HL am obtinut
                                                        ; codul in binar
7428    EB                      EX      DE,HL           ; se transfera codul in DE
7429    21 6C05                 LD      HL,SORTTOT      ; in HL adresa de inceput a totalurilor
742C    06 05                   LD      B,BCDLEN        ; in B numarul de octeti ai unei sume
742E    CD 7436                 CALL    ADDD            ; se executa HL=HL+DE*B,obtinindu-se in HL
                                                        ; adresa cautata
7431    FD E1                   POP     IY
7433    D1                      POP     DE
7434    C1                      POP     BC              ; se refac registrii salvati
7435    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-30

                        ;       Rutina ADDD
                        ;
                        ; Executa HL = HL + DE * B
                        ; Distruge: B,H,L
                        ;
7436                    ADDD:
7436    19                      ADD     HL,DE
7437    10 FD                   DJNZ    ADDD
7439    C9                      RET
                        ;
                        ;       Rutina ADDSORT
                        ;
                        ; Aduna pretul din DATBCD la totalul corespunzator codului din CODE.
                        ;
743A                    ADDSORT:
743A    CD 740F                 CALL    SORTADR         ; pe baza codului din CODE,formeaza in HL ad-
                                                        ; resa la care se gaseste totalul cautat
743D    E5                      PUSH    HL              ; salveaza adresa gasita
743E    CD 7705                 CALL    MOVTMP          ; transfera totalul selectat in buferul TMPBCD
7441    CD 755D                 CALL    ADDBCD          ; executa TMPBCD=TMPBCD+DATBCD
7444    D1                      POP     DE              ; reface in DE adresa totalului si
7445    CD 7712                 CALL    TMPMOV          ; transfera rezultatul adunarii din TMPBCD la
7448    C9                      RET                     ; aceasta adresa
                        ;
                        ;       Rutina SUBSORT
                        ;
                        ; Scade pretul din buferul DATBCD din totalul de sortiment corespunzator
                        ; codului din zona CODE
                        ;
7449                    SUBSORT:
7449    CD 740F                 CALL    SORTADR         ; formeaza in HL adresa totalului de sortiment
                                                        ; corespunzator codului din CODE
744C    E5                      PUSH    HL              ; se salveaza aceasta adresa
744D    CD 7705                 CALL    MOVTMP          ; se transfera totalul selectat in TMPBCD
7450    CD 7575                 CALL    SUBBCD          ; si se scade din el continutul lui DATBCD
7453    D1                      POP     DE              ; se reface adresa totalului selectat si
7454    CD 7712                 CALL    TMPMOV          ; se transfera aici valoarea din TMPBCD
7457    C9                      RET
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-31

                        ;
                        ;
                        ;       Rutina EMITBUYT
                        ;
                        ; Incrementeaza numaratorul bonurilor,aduna totalul clientului la totalul
                        ; zilei,afiseaza totalul clientului,imprima bonul clientului,iar la sfirsit
                        ; reinitializeaza totalul clientului cu 0.
                        ;
7458                    EMITBUYT:
7458    06 04                   LD      B,TCKNRLEN      ; se introduce in B lungimea numarului de bon,
745A    21 6EB6                 LD      HL,TICKNR+TCKNRLEN-1    ; in HL adresa ultimei cifre,
745D    CD 75E7                 CALL    EBCDINC         ; si se incrementeaza numarul bonului
7460    21 6C00                 LD      HL,DAYBCD       ; totalul zilei se transfera in
7463    CD 7705                 CALL    MOVTMP          ; buferul TMPBCD
7466    21 6DF9                 LD      HL,GUESTBCD     ; totalul clientului se transfera in
7469    11 6E08                 LD      DE,DATBCD       ; buferul DATBCD
746C    01 0005                 LD      BC,BCDLEN
746F    ED B0                   LDIR
7471    CD 755D                 CALL    ADDBCD          ; se aduna totalul clientului la totalul zilei
7474    11 6C00                 LD      DE,DAYBCD       ; iar rezultatul se transfera din TMPBCD la
7477    CD 7712                 CALL    TMPMOV          ; locul ei in DAYBCD
747A    21 6DF9                 LD      HL,GUESTBCD     ; totalul clientului se despacheteaza si
747D    11 6E1B                 LD      DE,PRTBUF+4     ; se introduce in buferul de imprimare
7480    CD 7666                 CALL    BCDCI           ; PRTBUF,
7483    CD 76F3                 CALL    DISPSUM         ; de unde se afiseaza
7486    21 6DF9                 LD      HL,GUESTBCD     ; totalul clientului se despacheteaza din nou
7489    11 6E93                 LD      DE,SUM          ; si se transfera in zona SUM din EDBUF
748C    CD 7666                 CALL    BCDCI
748F    0E 08                   LD      C,BUYMASK       ; in C masca de selectie
7491    CD 74E6                 CALL    PRTTICK         ; se imprima bonul clientului
7494    21 6DF9                 LD      HL,GUESTBCD     ; totalul clientului se reinitializeaza cu 0
7497    06 05                   LD      B,BCDLEN        ; ( in HL adresa,in B lungimea,in C valoarea
7499    0E 00                   LD      C,0             ; de introdus si se apeleaza rutina de
749B    CD 774A                 CALL    FILLBYTS        ; initializare )
749E    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE  1-32

                        ;
                        ;       Rutina EMITERAT
                        ;
                        ; Incrementeaza numaratorul bonurilor,scade totalul clientului din totalul
                        ; zilei,afiseaza totalul clientului,imprima bonul clientului,iar la sfirsit
                        ; reinitializeaza totalul clientului cu 0.
                        ;
749F                    EMITERAT:
749F  06 04                     LD      B,TCKNRLEN      ;se introduce in B lungimea numarului de bon,
74A1  21 6EB6                   LD      HL,TICKNR+TCKNRLEN-1    ; in HL adresa ultimei cifre,
74A4  CD 75E7                   CALL    EBCDINC         ;si se incrementeaza numarul bonului
74A7  21 6C00                   LD      HL,DAYBCD       ;totalul zilei se transfera in
74AA  CD 7705                   CALL    MOVTMP          ;buferul TMPBCD
74AD  21 6DF9                   LD      HL,GUESTBCD     ;totalul clientului se transfera in
74B0  11 6E08                   LD      DE,DATBCD       ;buferul DATBCD
74B3  01 0005                   LD      BC,BCDLEN
74B6  ED B0                     LDIR
74B8  CD 7575                   CALL    SUBBCD          ;se scade totalul clientului din totalul zilei
74BB  11 6C00                   LD      DE,DAYBCD       ;iar rezultatul se transfera din TMPBCD la
74BE  CD 7712                   CALL    TMPMOV          ;locul ei in DAYBCD
74C1  21 6DF9                   LD      HL,GUESTBCD     ;totalul clientului se despacheteaza si
74C4  11 6E1B                   LD      DE,PRTBUF+4     ;se introduce in buferul de imprimare
74C7  CD 7666                   CALL    BCDCI           ;PRTBUF,
74CA  CD 76F3                   CALL    DISPSUM         ;de unde se afiseaza
74CD  21 6DF9                   LD      HL,GUESTBCD     ;totalul clientului se despacheteaza din nou
74D0  11 6E93                   LD      DE,SUM          ;si se transfera in zona SUM din EDBUF
74D3  CD 7666                   CALL    BCDCI
74D6  0E 08                     LD      C,BUYMASK       ;in C masca de selectie
74D8  CD 74E6                   CALL    PRTTICK         ;se imprima bonul clientului
74DB  21 6DF9                   LD      HL,GUESTBCD     ;totalul clientului se reinitializeaza cu 0
74DE  06 05                     LD      B,BCDLEN        ;( in HL adresa,in B lungimea,in C valoarea
74E0  0E 00                     LD      C,0             ;de introdus si se apeleaza rutina de
74E2  CD 774A                   CALL    FILLBYTS        ;initializare )
74E5  C9                        RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-33

                        ;
                        ;       Rutina PRTTICK
                        ;
                        ; Imprima pe bonuri liniile din EDBUF selectate cu masca de selectie primita
                        ; de la rutina apelanta in C.Un caz special reprezinta bonul de tip sinteza.
                        ; In acest caz se apeleaza rutina SYNTTOTS care imprima toate totalurile de
                        ; sortiment fara a le mai introduce in PRTBUF.
                        ;
74E6                    PRTTICK:
74E6    06 0E                   LD      B,NREDLNS       ; in B numarul de linii din EDBUF
74E8    DD 21 6E3B              LD      IX,EDBUF+2      ; in IX adresa de inceput a textului
                                                        ; din prima linie
74EC    11 0011                 LD      DE,EDLLEN       ; in DE lungimea liniilor din EDBUF
74EF                    EDLINE:
74EF    DD 7E FF                LD      A,(IX-1)        ; octetul indicator al liniei se transfera in
74F2    A1                      AND     C               ; A si cu masca de selectie se testeaza daca
                                                        ; linia curenta face parte din bonul dorit
74F3    28 17                   JR      Z,NEXTEDLN      ; daca nu,salt pentru avans la urmatoarea linie
74F5    79                      LD      A,C             ; se testeaza daca bonul in curs este bon de
74F6    FE 01                   CP      SYNTMASK        ; tip sinteza
74F8    20 0C                   JR      NZ,NORMLINE     ; daca nu,salt la rind normal
74FA    DD 7E FE                LD      A,(IX-2)        ; daca da,atunci se testeaza daca este rindul
74FD    FE 86                   CP      86H             ; care contine zona SUM
74FF    20 05                   JR      NZ,NORMLINE     ; daca nu,salt la rind normal
7501                    SPEC:                           ; daca da,inseamna ca trebuiesc imprimate toate
7501    CD 7529                 CALL    SYNTTOTS        ; totalurile de sortiment
7504    18 06                   JR      NEXTEDLN        ; salt pentru avans la urmatoarea linie
7506                    NORMLINE:                       ; in cazul unei linii normale
7506    CD 7519                 CALL    TRANLINE        ; se transfera linia in buferul de imprimare
7509    CD 783A                 CALL    PRTLINE         ; si se imprima
750C                    NEXTEDLN:
750C    DD 19                   ADD     IX,DE           ; se pozitioneaza IX la inceputul urmatorului
750E    10 DF                   DJNZ    EDLINE          ; text si daca mai sint linii in EDBUF,atunci
                                                        ; salt inapoi
7510    21 03E8                 LD      HL,EDBFRE       ; la terminarea buferului EDBUF
7513    3E 14                   LD      A,EDBTIME       ; se suna soneria
7515    CD 77C3                 CALL    BEEP
7518    C9                      RET
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-34

                        ;
                        ;
                        ;       Rutina TRANLINE
                        ;
                        ; Transfera o linie din EDBUF in PRTBUF.Primeste de la rutina apelanta in IX
                        ; aderesa la care incepe linia de transferat.
                        ; Distruge: H,L
                        ; Registrii utilizati:
                        ;       HL- adresa din EDBUF
                        ;       DE- adresa din PRTBUF
                        ;       BC- numarul de octeti de transferat
                        ;
7519                    TRANLINE:
7519  C5                        PUSH    BC              ; se salveaza registrii utilizati
751A  D5                        PUSH    DE
751B  DD E5                     PUSH    IX              ; adresa liniei se transfera din IX
751D  E1                        POP     HL              ; in HL
751E  11 6E18                   LD      DE,PRTBUF+1     ; in DE adresa la care se transfera linia
7521  01 000F                   LD      BC,EDLLEN-2     ; in BC numarul de octeti de transferat
7524  ED B0                     LDIR
7526  D1                        POP     DE
7527  C1                        POP     BC              ; se refac registrii salvati
7528  C9                        RET
                                PAGE
```

```
                        ;
                        ;
                        ;       Rutina SYNTTOTS
                        ;
                        ; Imprima toate totalurile de sortiment,in cadrul bonului de tip sinteza.
                        ;
7529                    SYNTTOTS:
7529    D5                      PUSH    DE              ; se salveaza registrii utilizati
752A    C5                      PUSH    BC
752B    0E 00                   LD      C,0             ; se initializeaza zona CODE
752D    CD 739B                 CALL    IMPLCODE        ; cu codul 00
7530    06 64                   LD      B,100           ; se initializeaza numaratorul totalurilor
7532                    NSORTT:
7532    C5                      PUSH    BC              ; se salveaza numaratorul
7533    CD 740F                 CALL    SORTADR         ; determina adresa totalului corespunzator co-
7536    11 6E1B                 LD      DE,PRTBUF+4     ; dului din CODE si transfera totalul,despa-
7539    CD 7666                 CALL    BCDCI           ; chetat,in PRTBUF
753C    21 6E57                 LD      HL,CODE         ; codul din CODE se introduce in PRTBUF
753F    11 6E18                 LD      DE,PRTBUF+1     ; inaintea totalului
7542    01 0002                 LD      BC,SCLEN
7545    ED B0                   LDIR
7547    3E 0D                   LD      A,ASTER         ; se introduce "*" dupa total
7549    32 6E26                 LD      (PRTBUF+15),A
754C    CD 783A                 CALL    PRTLINE         ; se imprima linia formata
754F    06 02                   LD      B,SCLEN         ; se incrementeaza codul din CODE ( in B
7551    21 6E58                 LD      HL,CODE+SCLEN-1 ; lungimea,in HL adresa si cheama
7554    CD 75E7                 CALL    EBCDINC         ; rutina de incrementare )
7557    C1                      POP     BC              ; se reface numaratorul,se decrementeaza si
7558    10 D8                   DJNZ    NSORTT          ; daca nu am terminat,salt inapoi pentru
                                                        ; urmatorul total
755A    C1                      POP     BC
755B    D1                      POP     DE              ; se refac registrii salvati
755C    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-36

                        ;       Rutina ADDBCD
                        ;
                        ; Aduna continutul buferului TMPBCD cu continutul buferului DATBCD,rezultatul
                        ; formindu-se in TMPBCD.
                        ; Distruge: A
                        ; Registrii utilizati:
                        ;       DE- pentru adresarea locatiilor din TMPBCD
                        ;       HL- pentru adresarea locatiilor din DATBCD
                        ;       B - numarator
                        ;       A - rezultate partiale
7550                    ADDBCD:
7550    C5                      PUSH    BC
7551    D5                      PUSH    DE
7552    E5                      PUSH    HL
7553    11 6E07                 LD      DE,TMPBCD+BCDLEN-1      ; adresa ultimei locatii din TMPBCD
7556    21 6E0C                 LD      HL,DATBCD+BCDLEN-1      ; adresa ultimei locatii din DATBCD
7559    06 05                   LD      B,BCDLEN        ; numarul de octeti de adunat
755B    AF                      XOR     A               ; A = 0,CY = 0
755C                    ADDBYTE:
755C    1A                      LD      A,(DE)          ; octetul curent din TMPBCD se transfera in A
755D    8E                      ADC     A,(HL)          ; si se aduna cu octetul curent din DATBCD
755E    27                      DAA                     ; rezultatul se ajusteaza la 2 cifre BCD
755F    12                      LD      (DE),A          ; si se transfera in TMPBCD
7560    1B                      DEC     DE              ; se trece la urmatoarele locatii din TMPBCD
7561    2B                      DEC     HL              ; si din DATBCD
7562    10 F8                   DJNZ    ADDBYTE         ; daca au mai ramas octeti,salt inapoi
7564    E1                      POP     HL
7565    D1                      POP     DE
7566    C1                      POP     BC
7567    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-37


                        ;       Rutina SUBBCD
                        ;
                        ; Scade continutul buferului DATBCD din continutul buferului TMPBCD,rezultatul
                        ; formindu-se in TMPBCD.Indicatorul CY indica rutinei apelante daca rezultatul
                        ; scaderii este corect ( la rezultat negativ CY = 1 ).
                        ; Distruge: A
                        ; Registrii utilizati:
                        ;       DE- pentru adresarea locatiilor din TMPBCD
                        ;       HL- pentru adresarea locatiilor din DATBCD
                        ;       B - numarator
                        ;       A - rezultate partiale
7575                    SUBBCD:
7575    C5                      PUSH    BC
7576    D5                      PUSH    DE
7577    E5                      PUSH    HL
7578    11 6E07                 LD      DE,TMPBCD+BCDLEN-1      ; adresa ultimei locatii din TMPBCD
757B    21 6E0C                 LD      HL,DATBCD+BCDLEN-1      ; adresa ultimei locatii din DATBCD
757E    06 05                   LD      B,BCDLEN        ; numarul de octeti de adunat
7580    AF                      XOR     A               ; A = 0,CY = 0
7581                    SUBBYTE:
7581    1A                      LD      A,(DE)          ; octetul curent din TMPBCD se transfera in A
7582    9E                      SBC     A,(HL)          ; si se scade din el octetul curent din DATBCD
7583    27                      DAA                     ; rezultatul se ajusteaza la 2 cifre BCD
7584    12                      LD      (DE),A          ; si se transfera in TMPBCD
7585    1B                      DEC     DE              ; se trece la urmatoarele locatii din TMPBCD
7586    2B                      DEC     HL              ; si din DATBCD
7587    10 F8                   DJNZ    SUBBYTE         ; daca au mai ramas octeti,salt inapoi
7589    E1                      POP     HL
758A    D1                      POP     DE
758B    C1                      POP     BC
758C    C9                      RET
                                PAGE
```

```
                                ;
                                ;
                                ;       Rutina MULTBCD
                                ;
                                ; Este apelata din PRELPROC,daca indicatorul de inmultire este setat.
                                ; In momentul apelului avem in TMPBCD inmultitorul,iar in DATBCD deinmultitul.
                                ; Rutina trece inmultitorul in WORKBCD,aduce TMPBCD la 0 si executa adunarile
                                ; repetate,in functie de cifrele inmultitorului.La sfirsit rezultatul final
                                ; se trece din TMPBCD in DATBCD.
                                ; Distruge: A,B,C,D,E,H,L
                                ;
758D                            MULTBCD:
758D    11 6DFE                         LD      DE,WORKBCD      ; se transfera inmultitorul din TMPBCD
7590    CD 7712                         CALL    TMPMOV          ; in WORKBCD
7593    06 05                           LD      B,BCDLEN        ; buferul TMPBCD se initializeaza cu 0
7595    21 6E03                         LD      HL,TMPBCD       ; ( in B lungimea,in HL adresa,in C valoarea
7598    0E 00                           LD      C,0             ; si cheama rutina de initializare )
759A    CD 774A                         CALL    FILLBYTS
759D    06 05                           LD      B,BCDLEN        ; in B lungimea buferelor implicate
759F    0E 0A                           LD      C,BCDLEN*2      ; in C numarul de cifre ai operanzilor
75A1                            SHIFTS:
75A1    21 6E02                         LD      HL,WORKBCD+BCDLEN-1     ; cifrele din WORKBCD se deplaseaza
75A4    CD 75C7                         CALL    MUL10           ; cu o pozitie la stinga ( cea mai semnificati-
                                                                ; va intra in A )
75A7    5F                              LD      E,A             ; se salveaza
75A8    AF                              XOR     A               ; A = 0
75A9    21 6E07                         LD      HL,TMPBCD+BCDLEN -1     ; cifrele din TMPBCD se deplaseaza cu
75AC    CD 75C7                         CALL    MUL10           ; o pozitie la stinga ( in pozitia cea mai
                                                                ; putin semnificativa intra 0 )
75AF    1C                              INC     E
75B0                            MULT:
75B0    1D                              DEC     E               ; se aduna deinmultitul la rezultatul partial
75B1    28 05                           JR      Z,MULTENDT      ; de cite ori indica cifra cea mai semnifica-
75B3    CD 755D                         CALL    ADDBCD          ; tiva a inmultitorului
75B6    18 F8                           JR      MULT
75B8                            MULTENDT:
75B8    0D                              DEC     C               ; se decrementeaza numaratorul cifrelor
75B9    20 E6                           JR      NZ,SHIFTS       ; daca nu am terminat inmultirea,salt inapoi
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-39

75BB    21 6E03                 LD      HL,TMPBCD       ; rezultatul final al inmultirii
75BE    11 6E09                 LD      DE,DATBCD+1     ; se transfera din TMPBCD in DATBCD,cele mai
75C1    01 0004                 LD      BC,BCDLEN-1     ; putin semnificative 2 cifre pierzindu-se
75C4    ED B0                   LDIR                    ; ( ramin numai 2 cifre zecimale )
75C6    C9                      RET
                        ;
                        ;       Rutina MUL10
                        ;
                        ; Deplaseaza la stinga cu o pozitie ( inmulteste cu 10 ) numarul din zona
                        ; indicata de HL ( HL indica ultimul octet ).Cea mai semnificativa cifra trece
                        ; in A.
                        ; Distruge: A,H,L
                        ; Registrii utilizati:
                        ;       B - numarator al octetilor
                        ;       HL- pentru adresarea zonei
                        ;
75C7                    MUL10:
75C7    C5                      PUSH    BC
75C8                    M10:
75C8    ED 6F                   RLD                     ; se deplaseaza cifrele spre stinga
75CA    2B                      DEC     HL              ; ( cele care trec dintr-un octet in alt
75CB    10 FB                   DJNZ    M10             ; octet trec prin A )
75CD    C1                      POP     BC
75CE    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-40

                        ;       Rutina MULTOP
                        ;
                        ; Apelata in cazul actionarii tastei "*" dupa introducerea unui pret,rutina
                        ; analizeaza pretul introdus si daca este corect,il transfera in buferul
                        ; TMPBCD si seteaza indicatorul de inmultire.
75CF                    MULTOP:
75CF    F5                      PUSH    AF              ; se salveaza A
75D0    CD 75F2                 CALL    SYNTAN          ; se analizeaza pretul introdus si daca
75D3    20 0D                   JR      NZ,ENDMOP       ; este eronat,salt la sfirsitul rutinei
                                                        ; daca a fost corect,atunci in urma analizei
                                                        ; se gaseste in buferul EBCD
75D5    CD 7650                 CALL    EBCDBCD         ; pretul din EBCD in DATBCD
75D8    21 6E08                 LD      HL,DATBCD       ; iar din DATBCD in buferul TMPBCD
75DB    CD 7705                 CALL    MOVTMP
75DE    D9                      EXX
75DF    CB C3                   SET     0,E             ; si se seteaza indicatorul de inmultire
75E1    D9                      EXX
75E2                    ENDMOP:
75E2    CD 771F                 CALL    CLBUFDP         ; la sfirsitul rutinei se sterg KEYBUF si
75E5    F1                      POP     AF              ; LEDBUF si se reface A
75E6    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-41


                        ;
                        ;
                        ;       Rutina EBCDINC
                        ;
                        ; Incrementeaza numarul ( in BCD extins ) din zona de memorie indicata de HL
                        ; ( HL indica cifra cea mai putin semnificativa ).
                        ; Distruge: A,B,H,L
                        ; Registrii utilizati:
                        ;       B - numarul de octeti ai zonei de incrementat
                        ;       HL- pentru adresarea octetilor zonei
                        ;       A - utilizat in operatii
75E7                    EBCDINC:
75E7    34                      INC     (HL)            ; se incrementeaza cifra curenta a numarului
75E8    7E                      LD      A,(HL)
75E9    FE 0A                   CP      10              ; si se testeaza daca s-a ajuns la 10
75EB    C0                      RET     NZ              ; daca nu,operatia de incrementare s-a terminat
75EC    36 00                   LD      (HL),0          ; daca da,se schimba 10 cu 0 si
75EE    2B                      DEC     HL              ; se trece la cifra mai semnificativa
75EF    10 F6                   DJNZ    EBCDINC
75F1    C9                      RET
                                PAGE
```

```
                        ;
                        ;       Rutina SYNTAN
                        ;
                        ; Analizeaza pretul din KEYBUF si il transfera in EBCD.La sfirsit pozitioneaza
                        ; indicatorul Z pentru a indica rutinei apelante daca pretul a fost corect
                        ; sau nu.
                        ; Distruge: A,B,C,D,E,H,L,IX
                        ; Registrii utilizati:
                        ;       A - la diferite operatii
                        ;       B - numarator al locatiilor buferului KEYBUF
                        ;       C - numarator al cifrelor zecimale din EBCD
                        ;       D - indicator de eroare
                        ;       E - pentru salvari din A
                        ;       HL- pentru adresarea locatiilor buferului KEYBUF
                        ;       IX- pentru adresarea partii zecimale a buferului EBCD
                        ;
75F2                    SYNTAN:
75F2    06 0A                   LD      B,EBCDLEN       ; in B lungimea buferului EBCD,
75F4    21 6E0D                 LD      HL,EBCD         ; in HL adresa
75F7    0E 00                   LD      C,0             ; se initializeaza locatiile buferul cu 0
75F9    CD 774A                 CALL    FILLBYTS
75FC    21 6E29                 LD      HL,KEYBUF       ; in continuare HL va adresa locatiile din
75FF    06 08                   LD      B,BUFLEN        ; KEYBUF,in B lungimea buferului
7601    16 00                   LD      D,0             ; D va fi indicatorul de eroare
7603    CD 78C1                 CALL    SRCNBLK         ; se cauta in KEYBUF,de la stinga,prima locatie
                                                        ; care nu contine spatiu
7606    5F                      LD      E,A             ; se salveaza octetul citit in E si
7607    CB BF                   RES     7,A             ; se reseteaza bitul indicator de punct zecimal
7609    FE 20                   CP      BLANK           ; daca a ramas spatiu,inseamna ca pretul nu are
760B    28 17                   JR      Z,DECPART       ; parte intreaga,deci salt la analiza partii
                                                        ; zecimale
760D                    INTPART:                        ; daca nu,incepe analiza partii intregi
760D    CD 76E0                 CALL    STOREINT        ; prima cifra se memoreaza in EBCD
7610                    TINTEND:
7610    78                      LD      A,B             ; se testeaza daca numaratorul locatiilor lui
7611    B7                      OR      A               ; KEYBUF a ajuns la 0;daca da,analiza s-a
7612    28 2C                   JR      Z,FORMMES       ; terminat si salt la formarea mesajului
7614    7B                      LD      A,E             ; se reface octetul citit si
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-43

7615    CB 7F                   BIT     7,A             ; se testeaza bitul indicator al punctului
7617    20 0B                   JR      NZ,DECPART      ; daca este 1,atunci analiza partii intregi
                                                        ; s-a terminat si salt la partea zecimala
7619    5E                      LD      E,(HL)          ; daca a fost 0,se ia un nou octet din KEYBUF
761A    23                      INC     HL
761B    05                      DEC     B               ; se decrementeaza numaratorul octetilor
761C    7B                      LD      A,E             ; octetul citit se transfera in A
761D    CB BF                   RES     7,A             ; se reseteaza bitul indicator de punct zecimal
761F    CD 76E0                 CALL    STOREINT        ; si se memoreaza cifra in EBCD
7622    18 EC                   JR      TINTEND
7624                    DECPART:
7624    DD 21 6E15              LD      IX,EBCD+BUFLEN  ; IX va adresa partea zecimala din EBCD,iar
7628    0E 02                   LD      C,2             ; C va fi numaratorul cifrelor zecimale
762A                    NEXTDEC:
762A    5E                      LD      E,(HL)          ; se ia un nou octet din KEYBUF
762B    23                      INC     HL
762C    CB 7B                   BIT     7,E             ; se testeaza bitul indicator de punct zecimal
762E    28 04                   JR      Z,WITHOUTP      ; daca nu contine punct,salt
7630                    WITHP:                          ; daca da,inseamana ca sint mai multe puncte in
7630    16 01                   LD      D,1             ; pret,cea ce este eronat,deci se seteaza
7632    18 0C                   JR      FORMMES         ; indicatorul de eroare si salt la formare
7634                    WITHOUTP:                       ; mesaj
7634    79                      LD      A,C             ; se testeaza daca numaratorul cifrelor zecima-
7635    B7                      OR      A               ; le este 0;daca da,inseamna ca cifra nu face
                                                        ; parte din primele 2 cifre zecimale,deci nu
7636    28 06                   JR      Z,TDECEND       ; se retine
7638                    STOREDEC:
7638    DD 73 00                LD      (IX+0),E        ; in caz contrar se memoreaza cifra
763B    DD 23                   INC     IX
763D    0D                      DEC     C               ; si se decrementeaza numaratorul cifrelor
763E                    TDECEND:                        ; zecimale
763E    10 EA                   DJNZ    NEXTDEC         ; daca KEYBUF mai are cifre neanalizate,salt
                                                        ; inapoi
7640                    FORMMES:
7640    7A                      LD      A,D             ; indicatorul de eroare in A si
7641    B7                      OR      A               ; se pozitioneaza indicatorul Z
7642    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-44


                        ;
                        ;
                        ;       Rutina PRELPROC
                        ;
                        ; Impacheteaza pretul din buferul EBCD in buferul DATBCD si daca indicatorul
                        ; de inmultire este setat,atunci executa inmultirea.La sfirsitul rutinei
                        ; in DATBCD se gaseste numarul care trebuie adunat la totalul pe sortiment
                        ; si la totalul clientului.
                        ;
7643                    PRELPROC:
7643    CD 7650                 CALL    EBCDBCD         ; pretul din EBCD se impacheteaza in DATBCD
7646    D9                      EXX
7647    CB 43                   BIT     0,E             ; testeaza indicatorul de inmultire ( pozitio-
                                                        ; neaza indicatorul Z ),
7649    CB 83                   RES     0,E             ; dupa care il reseteaza ( Z nu mai este
                                                        ; afectat )
764B    D9                      EXX
764C    C4 758D                 CALL    NZ,MULTBCD      ; daca indicatorul a indicat inmultire,se
                                                        ; executa
764F    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-45

                        ;
                        ;
                        ;       Rutina EBCDBCD
                        ;
                        ; Pretul din buferul EBCD ( in BCD extins ) se impacheteaza si se transfera
                        ; in buferul DATBCD.
                        ; Distruge: A,B,D,E,H,L
                        ; Registrii utilizati:
                        ;       HL- adresa locatiei curente din EBCD
                        ;       DE- adresa locatiei curente din DATBCD
                        ;       B - numarator al octetilor din DATBCD
                        ;
7650                    EBCDBCD:
7650    21 6E0D                 LD      HL,EBCD         ; in HL adresa buferului EBCD
7653    11 6E08                 LD      DE,DATBCD       ; in DE adresa buferului DATBCD
7656    06 05                   LD      B,BCDLEN        ; in B numarul de octeti de format in DATBCD
7658                    PACKBYTE:
7658    7E                      LD      A,(HL)          ; prima cifra din cele doua care vor intra in
7659    23                      INC     HL              ; octetul curent a buferului DATBCD se
765A    07                      RLCA                    ; transfera in A pe bitii 7-4
765B    07                      RLCA
765C    07                      RLCA
765D    07                      RLCA
765E    ED 67                   RRD                     ; a doua cifra se transfera in A pe bitii 3-0
7660    12                      LD      (DE),A          ; octetul format se memoreaza in DATBCD
7661    23                      INC     HL
7662    13                      INC     DE
7663    10 F3                   DJNZ    PACKBYTE        ; daca nu am terminat,salt inapoi
7665    C9                      RET
                                PAGE
```

```
                        ;
                        ;
                        ;       Rutina BCDCI
                        ;
                        ; Despacheteaza numarul de la adresa indicata de HL in cod intern,la adresa
                        ; indicata de  DE,introduce punctul zecimal si inlocuieste zerourile
                        ; nesemnificative cu spatii.
                        ; Distruge: A,B,C,D,E,H,L
                        ; Registrii utilizati:
                        ;       HL- pentru adresarea zonei sursa in timpul despachetarii si a zonei
                        ;           ce contine numarul despachetat la introducerea punctului zecimal
                        ;           la inlocuirea zerourilor nesemnificative
                        ;       DE- pentru adresarea zonei destinatie in timpul despachetarii
                        ;
7666                    BCDCI:
7666    CD 7705                 CALL    MOVTMP          ; se transfera numarul de despachetat in
7669    21 6E03                 LD      HL,TMPBCD       ; TMPBCD si in HL adresa buferului
766C    06 05                   LD      B,BCDLEN        ; in B numarul de octeti de despachetat
766E    0E FF                   LD      C,0FFH
7670    AF                      XOR     A               ; A = 0 si indicatorii pozitionati
7671    D5                      PUSH    DE              ; se salveaza adresa la care se formeaza
                                                        ; numarul in cod intern
7672                    BCDCI1:
7672    ED 6F                   RLD                     ; cifra mai semnificativa a octetului curent
7674    12                      LD      (DE),A          ; din numarul de despachetat se transfera in
7675    13                      INC     DE              ; zona destinatie
7676    AF                      XOR     A               ; A = 0 si se reface cifra mai putin semnifi-
7677    ED 67                   RRD                     ; cativa in octetul curent din numarul de
7679    ED A0                   LDI                     ; despachetat si se transfera in zona
                                                        ; destinatie
767B    10 F5                   DJNZ    BCDCI1          ; daca nu am terminat,salt inapoi
767D    01 0002                 LD      BC,2            ; in BC numarul cifrelor zecimale,
7680    62                      LD      H,D             ; si aceste doua cifre se deplaseaza cu o
7681    6B                      LD      L,E             ; pozitie la dreapta pentru a face loc
7682    2B                      DEC     HL              ; punctului zecimal
7683    ED B8                   LDDR
7685    23                      INC     HL
7686    36 0A                   LD      (HL),POINT      ; se introduce punctul zecimal
```

```
MACRO-80 5.80   15-Jun-86     PAGE   1-47

7688    E1                          POP     HL              ; se reface adresa zonei destinatie
7689    06 07                       LD      B,BUFLEN-1      ; in B numarul maxim posibil de zerouri
                                                            ; nesemnificative de inlocuit
768B    2B                          DEC     HL
768C                        CHANGE:
768C    23                          INC     HL
768D    7E                          LD      A,(HL)          ; se testeaza daca octetul curent contine 0
768E    B7                          OR      A
768F    C0                          RET     NZ              ; daca nu,se revine in rutina apelanta
7690    36 20                       LD      (HL),BLANK      ; daca da,se inlocuieste cu spatiu si
7692    10 F8                       DJNZ    CHANGE          ; daca mai avem octeti de testat,salt inapoi
7694    C9                          RET
                                    PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-48

                        ;
                        ;
                        ;       Rutina STORENUM
                        ;
                        ; Introduce in buferul de tastatura KEYBUF cifra sau punctul primit in A.
                        ; In KEYBUF cifrele sint reprezentate in cod intern,iar punctul prin
                        ; setarea bitului cel mai semnificativ al octetului corespunzator.
                        ; Nu distruge nici un registru.
                        ; Registrii utilizati:
                        ;       A - cifra sau punctul ( de la rutina apelanta )
                        ;       DE- pentru adresarea locatiilor buferului
                        ;       HL-             - I I -
                        ;       BC- numarator la deplasarea continutului cu o pozitie la stinga
                        ;
7695                    STORENUM:
7695    C5              PUSH    BC              ; se salveaza registrii utilizati
7696    D5              PUSH    DE
7697    E5              PUSH    HL
7698    FE 0A           CP      POINT           ; trebuie introdus PUNCT ?
769A    28 0E           JR      Z,STPOINT       ; daca da,salt la secventa de memorare PUNCT
769C                    STNUM:                  ; daca nu,trebuie introdus o cifra
769C    11 6E29         LD      DE,KEYBUF       ; continutul buferului se deplaseaza cu o
769F    21 6E2A         LD      HL,KEYBUF+1     ; pozitie la stinga ( cel mai semnificativ
76A2    01 0007         LD      BC,BUFLEN-1     ; octet se pierde ); la sfirsitul deplasarii
76A5    ED B0           LDIR                    ; in DE adresa celui mai putin semnificativ
76A7    12              LD      (DE),A          ; octet,in care se introduce noua cifra si
76A8    18 05           JR      ENDST           ; salt la sfirsit
76AA                    STPOINT:
76AA    21 6E30         LD      HL,KEYBUF+BUFLEN-1      ; daca punct,se seteaza bitul 7 al
76AD    CB FE           SET     7,(HL)          ; celui mai putin semnificativ octet
76AF                    ENDST:
76AF    E1              POP     HL
76B0    D1              POP     DE
76B1    C1              POP     BC              ; se refac registrii salvati
76B2    C9              RET
                        PAGE
```

```
                        ;       Rutina DISPNUM
                        ;
                        ; Introduce in buferul de afisaj LEDBUF cifra sau punctul primit in A.
                        ; In LEDBUF cifrele sint reprezentate in cod de 7 segmente,iar punctul prin
                        ; resetarea bitului cel mai semnificativ al octetului corespunzator,deci in
                        ; forma in care se trimit la portul LEDPORT.
                        ; Nu distruge nici un registru.
                        ; Registrii utilizati:
                        ;       A - cifra sau punctul ( de la rutina apelanta )
                        ;       DE,HL- pentru adresarea locatiilor buferului
                        ;       BC- numarator la deplasarea continutului cu o pozitie la stinga
                        ;           si la determinarea adresei codului de 7 segmente al cifrei noi
76B3                    DISPNUM:
76B3    F5              PUSH    AF              ; se salveaza registrii utilizati
76B4    C5              PUSH    BC
76B5    D5              PUSH    DE
76B6    E5              PUSH    HL
76B7    FE 0A           CP      POINT           ; trebuie introdus PUNCT ?
76B9    28 14           JR      Z,DPPOINT       ; daca da,salt la secventa de memorare PUNCT
76BB                    DPNUM:                  ; daca nu,trebuie introdus o cifra
76BB    11 6E31         LD      DE,LEDBUF       ; continutul buferului se deplaseaza cu o
76BE    21 6E32         LD      HL,LEDBUF+1     ; pozitie la stinga ( cel mai semnificativ
76C1    01 0007         LD      BC,BUFLEN-1     ; octet se pierde ); la sfirsitul deplasarii
76C4    ED B0           LDIR                    ; in DE adresa celui mai putin semnificativ
                                                ; octet,iar BC = 0
76C6    21 796C         LD      HL,LEDGEN       ; in HL adresa generatorului de cifre pe 7
76C9    4F              LD      C,A             ; segmente,iar in BC cifra ( B = 0 )
76CA    09              ADD     HL,BC           ; se calculeaza adresa codului cifrei ( in HL )
76CB    ED A0           LDI                     ; si codul se transfera in LEDBUF
76CD    18 05           JR      ENDDP           ; salt la sfirsit
76CF                    DPPOINT:
76CF    21 6E38         LD      HL,LEDBUF+BUFLEN-1      ; daca punct,se reseteaza bitul 7 al
76D2    CB BE           RES     7,(HL)          ; celui mai putin semnificativ octet
76D4    E1              ENDDP:  POP     HL
76D5    D1              POP     DE
76D6    C1              POP     BC
76D7    F1              POP     AF              ; se refac registrii salvati
76D8    C9              RET
```

```
                        ;
                        ;
                        ;       Rutina STORDISP
                        ;
                        ; Introduce cifra sau punctul primit de la rutina apelanta prin A in buferele
                        ; KEYBUF si LEDBUF.
                        ; Nu distruge nici un registru.
76D9                    STORDISP:
76D9    CD 7695                 CALL    STORENUM        ; in KEYBUF
76DC    CD 76B3                 CALL    DISPNUM         ; in LEDBUF
76DF    C9                      RET
                        ;
                        ;
                        ;       Rutina STOREINT
                        ;
                        ; Memoreaza cifra primita de la rutina apelanta prin A in EBCD,in pozitia cea
                        ; mai putin semnificativa a intregilor.
                        ; Nu distruge nici un registru.
                        ; Registrii utilizati:
                        ;       A - contine cifra de introdus in EBCD
                        ;       HL- indica adresa sursa la deplasarea spre stinga
                        ;       DE- indica adresa destinatie la deplasare
                        ;       BC- numarator la deplasare
                        ;
76E0                    STOREINT:
76E0    C5                      PUSH    BC              ; se salveaza registrii utilizati
76E1    D5                      PUSH    DE
76E2    E5                      PUSH    HL
76E3    11 6E0D                 LD      DE,EBCD         ; cifrele introduse anterior in partea intreaga
76E6    21 6E0E                 LD      HL,EBCD+1       ; a buferului se deplaseaza cu o pozitie la
76E9    01 0007                 LD      BC,EBCDLEN-3    ; stinga ( cea mai semnificativa se pierde )
76EC    ED B0                   LDIR
76EE    12                      LD      (DE),A          ; se introduce noua cifra
76EF    E1                      POP     HL
76F0    D1                      POP     DE
76F1    C1                      POP     BC              ; se refac registrii salvati
76F2    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-51
                                ;
                                ;
                                ;SUBRUTINA: Rutina DISPSUM
                                ;
                                ; Transfera in buferul de afisaj LEDBUF numarul continut de buferul PRTBUF.
                                ; Distruge: A,B,H,L
                                ; Registrii utilizati:
                                ;       HL- pentru adresarea buferului PRTBUF
                                ;       B - numarator
                                ;       A - la transmiterea cifrei de afisat rutinei DISPNUM
                                ;
76F3                            DISPSUM:
76F3    21 6E1B                         LD      HL,PRTBUF+4     ; adresa la care incepe numarul de afisat
76F6    06 0B                           LD      B,BUFLEN+2+1    ; in B numarul de octeti
76F8    CD 78C1                         CALL    SRCNBLK         ; se cauta primul octet care nu contine spatiu
76FB    2B                              DEC     HL              ; ( la revenirea din SRCNBLK HL indica deja
76FC    04                              INC     B               ; octetul urmator,de aceea sr decrementeaza )
76FD                            DISP1:
76FD    7E                              LD      A,(HL)          ; se transfera cifra curenta a numarului in A
76FE    CD 76B3                         CALL    DISPNUM         ; si se introduce in buferul de afisaj
7701    23                              INC     HL
7702    10 F9                           DJNZ    DISP1           ; daca mai sint cifre de transferat,salt inapoi
7704    C9                              RET
                                        PAGE
```

```
                        ;
                        ;
                        ;       Rutina MOVTMP
                        ;
                        ; Transfera continutul zonei indicate de HL in TMPBCD.
                        ; Distruge: H,L
                        ; Registrii utilizati:
                        ;       HL- pentru adresarea zonei sursa
                        ;       DE- pentru adresarea zonei destinatie ( TMPBCD )
                        ;       BC- numarul de octeti de transferat
7705                    MOVTMP:
7705  C5                        PUSH    BC              ; se salveaza registrii utilizati
7706  D5                        PUSH    DE
7707  11 6E03                   LD      DE,TMPBCD       ; in DE adresa bufferului TMPBCD,iar
770A  01 0005                   LD      BC,BCDLEN       ; in BC lungimea bufferului
770D  ED B0                     LDIR                    ; se executa transferul
770F  D1                        POP     DE
7710  C1                        POP     BC              ; se refac registrii salvati
7711  C9                        RET
                        ;
                        ;
                        ;       Rutina TMPMOV
                        ;
                        ; Transfera continutul bufferului TMPBCD la adresa indicata de DE
                        ; Distruge: D,E
                        ; Registrii utilizati:
                        ;       HL- pentru adresarea zonei sursa ( TMPBCD )
                        ;       DE- pentru adresarea zonei destinatie
                        ;       BC- numarul de octeti de transferat
7712                    TMPMOV:
7712  C5                        PUSH    BC              ; se salveaza registrii utilizati
7713  E5                        PUSH    HL
7714  21 6E03                   LD      HL,TMPBCD       ; in HL adresa,iar
7717  01 0005                   LD      BC,BCDLEN       ; in BC lungimea bufferului TMPBCD
771A  ED B0                     LDIR                    ; se executa transferul
771C  E1                        POP     HL
771D  C1                        POP     BC              ; se refac registrii salvati
771E  C9                        RET
```

```
MACRO-80 5.80   15-Jun-86        PAGE    1-53

                        ;
                        ;       Rutina CLBUFDP
                        ;
                        ; Sterge buferele KEYBUF si LEDBUF,introducind in ele spatii.
                        ; Nu distruge nici un registru.
771F                    CLBUFDP:
771F    CD 7726                 CALL    CLKEYBUF        ; se sterge KEYBUF
7722    CD 7735                 CALL    CLDISP          ; se sterge LEDBUF
7725    C9                      RET
                        ;
                        ;
                        ;       Rutina CLKEYBUF
                        ;
                        ; Sterge buferul KEYBUF,introducind spatii.
                        ; Nu distruge nici un registru.
                        ; Registrii utilizati:
                        ;       HL- pentru adresarea buferului
                        ;       B - ca numarator al locatiilor buferului
                        ;       C - contine valoarea de introdus
                        ;
7726                    CLKEYBUF:
7726    C5                      PUSH    BC              ; se salveaza registrii utilizati
7727    E5                      PUSH    HL
7728    06 08                   LD      B,BUFLEN        ; in B lungimea buferului
772A    21 6E29                 LD      HL,KEYBUF       ; in HL adresa de inceput
772D    0E 20                   LD      C,BLANK         ; in C valoarea de introdus,adica spatiu
772F    CD 774A                 CALL    FILLBYTS        ; si se apeleaza rutina de initializare
7732    E1                      POP     HL
7733    C1                      POP     BC              ; se refac registrii utilizati
7734    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-54

                        ;       Rutina CLDISP
                        ;
                        ; Sterge,introducind spatii,buferul LEDBUF de afisaj.
                        ; Nu distruge nici un registru.
                        ; Registrii utilizati:
                        ;       HL- pentru adresarea buferului
                        ;       B - lungimea buferului
                        ;       C - contine valoarea ce se introuce in bufer
7735                    CLDISP:
7735    C5                      PUSH    BC              ; se salveaza registrii utilizati
7736    E5                      PUSH    HL
7737    06 08                   LD      B,BUFLEN        ; in B lungimea buferului
7739    21 6E31                 LD      HL,LEDBUF       ; in HL adresa de inceput
773C    0E FF                   LD      C,BLKDISP       ; in C spatiu de afisaj ( segmentele stinse )
773E    CD 774A                 CALL    FILLBYTS        ; se apeleaza rutina de initializare
7741    E1                      POP     HL
7742    C1                      POP     BC              ; se refac registrii salvati
7743    C9                      RET
                        ;
                        ;       Rutina CLEARCOD
                        ;
                        ; Sterge buferele KEYBUF si LEDBUF si reinitializeaza numaratorul cifrelor
                        ; codului.Distruge: B
                        ; Registru utilizat:B - numarator al cifrelor codului
7744                    CLEARCOD:
7744    CD 771F                 CALL    CLBUFDP         ; se sterg KEYBUF si LEDBUF
7747    06 03                   LD      B,SCLEN+1       ; se reinitializeaza numaratorul
7749    C9                      RET
                        ;
                        ;       Rutina FILLBYTS
                        ;
                        ; Initializeaza o zona de memorie ce incepe la adresa data de HL,avind lungimea
                        ; indicata de B,cu valoarea continuta de C.Distruge: B,H,L
774A                    FILLBYTS:
774A    71                      LD      (HL),C          ; se transfera continutul lui C in locatia
774B    23                      INC     HL              ; curenta si se trece la urmatoarea locatie
774C    10 FC                   DJNZ    FILLBYTS
774E    C9                      RET
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-55

                ;
                ;       Rutina EXPAND
                ;
                ; Transfera mesajele din EPROM ( MESSAGE ) in RAM ( EDBUF ) expandind prin
                ; completare cu blancuri fiecare linie pina la lungimea de 15 caractere.
                ; Distruge: A,B,C,D,E,H,L
                ; Registrii utilizati:
                ;       HL- pentru adresarea zonei EDBUF
                ;       DE- pentru adresarea zonei MESSAGE
                ;       C - numarul de linii de transferat
                ;       B - numarul de octeti ai unei linii
                ;
774F            EXPAND:
774F  0E 0E             LD      C,NREDLNS       ; in C numarul de linii de transferat
7751  11 78F9           LD      DE,MESSAGE      ; in DE adresa de inceput a zonei MESSAGE
7754  21 6E39           LD      HL,EDBUF        ; in HL adresa de inceput a zonei EDBUF
7757  1A               LD      A,(DE)          ; se ia primul octet din MESSAGE
7758            EXPLINE:
7758  06 11             LD      B,EDLLEN        ; in B lungimea unei linii
775A            EXPCHRS:
775A  77               LD      (HL),A          ; octetul curent se transfera in EDBUF
775B  13               INC     DE
775C  23               INC     HL
775D  05               DEC     B               ; se decrementeaza numaratorul octetilor
775E  1A               LD      A,(DE)          ; se ia urmatorul octet din MESSAGE
775F  CB 7F            BIT     7,A             ; si se testeaza daca bitul 7 este 1
7761  28 F7            JR      Z,EXPCHRS       ; daca nu,salt inapoi la transfer
7763            EXPSPAC:                        ; daca da,urmeaza expandarea cu spatii
7763  36 20            LD      (HL),BLANK      ; in pozitiile ramase se introduc spatii
7765  23               INC     HL
7766  10 FB            DJNZ    EXPSPAC
7768  0D               DEC     C               ; se decrementeaza numaratorul liniilor si
7769  20 ED            JR      NZ,EXPLINE      ; daca mai sint linii,salt inapoi
776B  21 6E9E           LD      HL,SUM+11       ; dupa expandare se introduce "*" in EDBUF
776E  36 0D             LD      (HL),ASTER
7770  C9                RET
                        PAGE
```

```
                        ;
                        ;
                        ;       Rutina INKEY
                        ;
                        ; Citeste o tasta de la tastatura.
                        ;
                        ;
7771                    INKEY:
7771  E5                        PUSH    HL              ; se salveaza registrii utilizati
7772  C5                        PUSH    BC
7773  ED 5F                     LD      A,R             ; se genereaza aleator un numar intre 0 si 7
7775  E6 07                     AND     7               ; pentru a incepe testarea liniilor
7777  87                        ADD     A,A             ; pe baza numarului generat se selecteaza din
7778  01 77B3                   LD      BC,KEYTAB       ; KEYTAB masca liniei ( se transfera in B ) si
777B  6F                        LD      L,A             ; valoarea codului daca linia ar fi activa
777C  26 00                     LD      H,0             ; ( se transfera in L )
777E  09                        ADD     HL,BC
777F  46                        LD      B,(HL)
7780  23                        INC     HL
7781  6E                        LD      L,(HL)
7782  0E 90                     LD      C,SYSIN         ; in C adresa portului de intrari
7784                    NXTLIN:
7784  ED 78                     IN      A,(C)           ; se citeste portul
7786  2F                        CPL                     ; se complementeaza valoarea citita si
7787  E6 0C                     AND     MASK1           ; se selecteaza liniile KYBD
7789  28 1E                     JR      Z,NOKEY         ; daca sint inactive,salt la avans de linie
778B  06 03                     LD      B,SHIFTM        ; daca sint active,se cadreaza rezultatul la
778D                    SHIFTA:                         ; dreapta prin deplasari repetate,numarul de
778D  05                        DEC     B               ; deplasari fiind controlat de registrul B
778E  28 04                     JR      Z,TSHIFT
7790  CB 3F                     SRL     A
7792  18 F9                     JR      SHIFTA
7794                    TSHIFT:                         ; se testeaza daca nu sint actionate ambele
7794  FE 03                     CP      3               ; taste de pe linie
7796  20 01                     JR      NZ,OTASTA       ; daca nu,salt
7798  3D                        DEC     A               ; daca da,una se elimina
7799                    OTASTA:                         ; se formeaza codul tastei adunind la valoarea
7799  85                        ADD     A,L             ; citita codul corespunzator liniei si se
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-57

779A    F5                      PUSH    AF              ; salveaza valoarea formata
779B    01 0090                 LD      BC,SYSIN
779E                    ASTEPT:                         ; in programul apelant se revine numai dupa ce
779E    ED 78                   IN      A,(C)           ; se ridica tasta
77A0    2F                      CPL
77A1    E6 0C                   AND     MASK1
77A3    20 F9                   JR      NZ,ASTEPT
77A5    F1                      POP     AF              ; se refac codul tastei si registrii utilizati
77A6    C1                      POP     BC
77A7    E1                      POP     HL
77A8    C9                      RET
77A9                    NOKEY:                          ; daca tastele liniei curente nu sint active
77A9    2C                      INC     L               ; se trece la urmatoarea linie,shimbind
77AA    2C                      INC     L               ; codul liniei curente si masca
77AB    CB 00                   RLC     B
77AD    38 D5                   JR      C,NXTLIN        ; daca nu mai sint linii netestate,se revine
77AF    2E FF                   LD      L,-1            ; la prima din dreapta
77B1    18 D1                   JR      NXTLIN
                        ;
77B3                    KEYTAB:
77B3    FE FF                   DB      11111110B,0FFH
77B5    FD 01                   DB      11111101B,01H
77B7    FB 03                   DB      11111011B,03H
77B9    F7 05                   DB      11110111B,05H
77BB    EF 07                   DB      11101111B,07H
77BD    DF 09                   DB      11011111B,09H
77BF    BF 0B                   DB      10111111B,0BH
77C1    7F 0D                   DB      01111111B,0DH
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-58

                        ;
                        ;       Rutina BEEP
                        ;
                        ; Este rutina generatoare de sunet.
                        ; Distruge: A,H,L
                        ; Registrii utilizati:
                        ;       HL- contine frecventa in Hz
                        ;       A - contine un numar proportional cu durata D
                        ;
77C3                    BEEP:
77C3    C5                      PUSH    BC              ; se salveaza registrii
77C4    D5                      PUSH    DE
77C5    E5                      PUSH    HL
77C6    37                      SCF                     ; indicatorul CY se pozitioneaza pe 1,
77C7    3F                      CCF                     ; apoi se complementeaza
                        ;
                        ; TIME = D * 128
                        ; ( se calibreaza durata tonului )
77C8    1F                      RRA
77C9    67                      LD      H,A
77CA    3E 00                   LD      A,0
77CC    1F                      RRA
77CD    6F                      LD      L,A             ; HL = A * 128
                        ;
                        ; NCYCLE = ONEHZ / F
                        ; ( se calculeaza numarul de temporizari elementare )
77CE    E3                      EX      (SP),HL         ; HL = F ,(SP) = TIME
77CF    11 61A8                 LD      DE,ONEHZ
77D2    EB                      EX      DE,HL
77D3    CD 7805                 CALL    COMPUTE         ; se apeleaza rutina de impartire care retur-
                                                        ; neaza citul rotunjit in HL
                        ; NRIMP = TIME / NCYCLE
                        ; ( se calculeaza numarul de impulsuri de frecventa F ce vor fi generate pentru
                        ; a se obtine durata D )
77D6    E3                      EX      (SP),HL         ; HL = TIME, (SP) = NCYCLE
77D7    D1                      POP     DE              ; DE = NCYCLE
77D8    D5                      PUSH    DE
```

```
77D9    CD 7821                 CALL    DIVIDE          ; NRIMP  = TIME / NCYCLE,NRIMP in BC
77DC    03                      INC     BC              ; se evita BC = 0
                        ;
                        ; Se emite sunetul
77DD    D1                      POP     DE
77DE    CD 77E4                 CALL    SING            ; se emit NRIMP ( BC ) impulsuri cu durata
                                                        ; data de NCYCLE ( DE )
77E1    D1                      POP     DE
77E2    C1                      POP     BC              ; se refac registrii salvati
77E3    C9                      RET
                        ;
                        ;       Rutina SING
                        ;
                        ; Subrutina de emitere a sunetului.Primeste in BC numarul de impulsuri de emis
                        ; ( NRIMP ),in DE durata unui impuls ( NCYCLE ).
                        ; In dreptul instructiunilor numarul de stari necesare executiei.
                        ;
77E4                    SING:
77E4    D5                      PUSH    DE              ; se salveaza NCYCLE ,
77E5                    SING1:
77E5    3A 6F27                 LD      A,(WITNESS)     ; (13)  se foloseste celula martor a portului
                                                        ; de iesire
77E8    CB FF                   SET     BEEPBIT,A       ; (8)
77EA    D1                      POP     DE              ; (10)
77EB    B5                      PUSH    DE              ; (10)
77EC    D3 90                   OUT     (SYSOUT),A      ; (11)
77EE    CD 7817                 CALL    DELAYO          ; 17+DE*50+10
77F1    3A 6F27                 LD      A,(WITNESS)     ; (13)
77F4    CB BF                   RES     BEEPBIT,A       ; (8)
77F6    D1                      POP     DE              ; (10)
77F7    D5                      PUSH    DE              ; (10)
77F8    D3 90                   OUT     (SYSOUT),A      ; (11)
77FA    CD 7817                 CALL    DELAYO          ; 17+DE*50+10
77FD    0B                      DEC     BC              ; (6) NRIMP = NRIMP - 1
77FE    79                      LD      A,C             ; (4)
77FF    B0                      OR      B               ; (4)
7800    C2 77E5                 JP      NZ,SING1        ; (10)
7803    D1                      POP     DE
7804    C9                      RET
```

```
                        ;       Rutina COMPUTE
                        ;
                        ; Rutina de impartire cu rotunjirea citului.Primeste de la rutina apelanta
                        ; in HL deimpartitul,iar in DE impartitorul si returneaza in HL citul
                        ; rotunjit.
7805                    COMPUTE:
7805  CD 7821                   CALL    DIVIDE
7808  C5                        PUSH    BC              ; se salveaza citul nerotunjit
7809  29                        ADD     HL,HL           ; se dubleaza restul
780A  CD 7811                   CALL    COMP            ; daca restul dublat este mai mic decit
780D  E1                        POP     HL              ; impartitul,atunci citul nu se schimba
780E  D8                        RET     C
780F  23                        INC     HL              ; altfel,el este incrementat,rotunjindu-se
7810  C9                        RET                     ; rezultatul
                        ;
                        ;       Rutina COMP
                        ;
                        ; Compara registrii DE si HL.Indicatorul CY setat indica DE > HL.
7811                    COMP:
7811  B7                        OR      A               ; se reseteaza CY
7812  E5                        PUSH    HL
7813  ED 52                     SBC     HL,DE           ; CY = 1,daca DE > HL
7815  E1                        POP     HL
7816  C9                        RET
                        ;
                        ;       Rutina DELAYO
                        ;
                        ; Asigura decrementarea variabilei NCYCLE de la o valoare initiala la 0,
                        ; astfel incit fiecare iteratie sa dureze 50 tacti procesor.
                        ;
7817                    DELAYO:
7817  1B                        DEC     DE              ; (6)
7818  18 00                     JR      DELAY1          ; (12) salt de temporizare
781A  18 00             DELAY1: JR      DELAY2          ; (12) salt de temporizare
781C  7B                DELAY2: LD      A,E             ; (4) se testeaza daca DE = 0
781D  B2                        OR      D               ; (4)
781E  20 F7                     JR      NZ,DELAYO       ; (12) daca nu,salt inapoi
7820  C9                        RET
```

```
MACRO-80 5.80   15-Jun-86    PAGE    1-61

                        ;       Rutina DIVIDE
                        ;
                        ; Calculeaza NRIMP = TIME / NCYCLE
                        ;
7821                    DIVIDE:
7821    7C              LD      A,H             ; se transfera deinpartitul in registrii de
7822    4D              LD      C,L             ; lucru A si C
7823    21 0000         LD      HL,0            ; se initializeaza restul
7826    06 10           LD      B,10H
                        ;
                        ; shift left
                        ;
7828                    DIVIDO:
7828    CB 11           RL      C
782A    17              RLA
782B    ED 6A           ADC     HL,HL
                        ;
                        ; divint
                        ;
782D    ED 52           SBC     HL,DE
782F    30 01           JR      NC,DIVID1
7831    19              ADD     HL,DE
7832                    DIVID1:
7832    3F              CCF
                        ;
                        ; N = N - 1
                        ;
7833    10 F3           DJNZ    DIVIDO
                        ;
                        ; shift left
                        ;
7835    CB 11           RL      C
7837    17              RLA
7838    47              LD      B,A             ; octetul superior al citului se transfera
7839    C9              RET                     ; in A
                        PAGE
```

```
                        ;
                        ;
                        ;       Rutina PRTLINE
                        ;
                        ; Imprima continutul buferului PRTBUF pe ambele bonuri,iar la terminarea
                        ; imprimarii sterge PRTBUF,intoducind spatii
                        ;
783A                    PRTLINE:
783A F5                         PUSH    AF              ; se salveaza registrii
783B C5                         PUSH    BC
783C D5                         PUSH    DE
783D E5                         PUSH    HL
783E CD 78B1                    CALL    MOTON           ; se comanda pornirea motorului imprimantei
7841                    WAITPOZ:
7841 DB 90                      IN      A,(SYSIN)       ; se asteapta trecerea din 1 in 0 a semnalului
7843 CB 7F                      BIT     CARLIMIT,A      ; CARLIMIT,cea ce indica atingerea pozitiei in
7845 20 FA                      JR      NZ,WAITPOZ      ; care poate sa inceapa imprimarea
7847 1E 70                      LD      E,INERTDEL      ; pentru a elimina inertia de la inceputul cursei
7849 CD 78A6                    CALL    DELAY           ; se executa o intirziere
784C CD 7869                    CALL    PRT18C          ; se imprima PRTBUF pe bonul clientului
784F 1E 80                      LD      E,BTWTDEL       ; intirziere pentru ca capul de imprimare sa
7851 CD 78A6                    CALL    DELAY           ; ajunga la bonul de control
7854 CD 7869                    CALL    PRT18C          ; se imprima PRTBUF pe bonul de control iar la
7857 06 12                      LD      B,PRTLEN        ; sfirsit se reinitializeaza cu spatii
7859 21 6E17                    LD      HL,PRTBUF       ; ( in B lungimea zonei,in HL adresa
785C 0E 20                      LD      C,BLANK         ; in C octetul de introdus si
785E CD 774A                    CALL    FILLBYTS        ; se cheama rutina ce umple zona specificata )
7861 CD 78B6                    CALL    MOTOFF          ; se comanda oprirea motorului imprimantei
7864 E1                         POP     HL              ; se refac registrii salvati
7865 D1                         POP     DE
7866 C1                         POP     BC
7867 F1                         POP     AF
7868 C9                         RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-63

                        ;
                        ;
                        ;       Rutina PRT18C
                        ;
                        ; Imprima continutul buferului PRTBUF ( 18 caractere ) pe bonul la inceputul
                        ; caruia se afla capul de imprimare ( imprimare de la dreapta la stinga ).
                        ; Distruge: B,E,IY,D
                        ; Registrii utilizati:
                        ;       B - numaratorul caracterelor
                        ;       IY- adreseaza locatia curenta din PRTBUF
                        ;       E - pentru a transmite caracterul de tiparit  rutinei PRTCHAR
                        ;           si ca parametru  pentru rutina DELAY
7869                    PRT18C:
7869    FD 21 6E28              LD      IY,PRTBUF+17    ; se initializeaza IY cu adresa ultimei locatii
786D    06 12                   LD      B,PRTLEN        ; din PRTBUF,iar B cu lungimea buferului
786F                    PCHAR:
786F    FD 5E 00                LD      E,(IY+0)        ; se transfera caracterul curent din PRTBUF in E
7872    CD 787F                 CALL    PRTCHAR         ; si se apeleaza rutina de imprimare caracter
7875    1E 05                   LD      E,BTWCHDEL      ; in E parametrul de intirziere
7877    CD 78A6                 CALL    DELAY           ; intirziere de 1,5 ms intre caractere
787A    FD 2B                   DEC     IY              ; se trece la urmatoarea locatie din PRTBUF
787C    10 F1                   DJNZ    PCHAR           ; si daca mai sint caractere de tiparit,salt inapoi
787E    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE  1-64
                        ;
                        ;
                        ;       Rutina PRTCHAR
                        ;
                        ; Imprima caracterul primit de la rutina apelanta in registrul E.
                        ; Distruge: A,D,H,L
                        ; Registrii utilizati:
                        ;       A - pentru a transmite coloana curenta din caracter rutinei de imprimare
                        ;           a unei coloane
                        ;       B - indica rutinei ADDD numarul de adunari de efectuat si  numara coloanele
                        ;           de tiparit ale caracterului
                        ;       HL- pentru a adresa locatiile ce contin coloanele caracterului
                        ;       E - transmite caracterul de tiparit
                        ;       D - pentru a avea codul caracterului pe 16 biti
787F                    PRTCHAR:
787F    C5                      PUSH    BC              ; se salveaza registrii necesari rutinei apelante
7880    21 7976                 LD      HL,PRTGEN       ; se incarca in HL adresa generatorului de caractere
7883    06 05                   LD      B,5             ; si se aduna la el de 5 ori codul caracterului
7885    16 00                   LD      D,0             ; ( pe 16 biti in DE ),obtinindu-se in HL
7887    CD 7436                 CALL    ADDD            ; adresa coloanelor caracterului de tiparit
788A    06 05                   LD      B,5             ; se initializeaza B ca numarator al coloanelor
788C                    PCOL:
788C    7E                      LD      A,(HL)          ; se transfera coloana curenta in A
788D    CD 7895                 CALL    PRTCOL          ; si se imprima
7890    23                      INC     HL
7891    10 F9                   DJNZ    PCOL            ; daca mai sint coloane de tiparit,salt inapoi
7893    C1                      POP     BC              ; se refac registrii salvati
7894    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-65

                                ;
                                ;       Rutina PRTCOL
                                ;
                                ; Imprima o coloana dintr-un caracter,cu intirzierile corespunzatoare.
                                ; Rutina apelanta transmite coloana prin A.
                                ; Distruge: A,D,E
                                ; Registrii utilizati:
                                ;       A - pentru a transmite comenzile la portul acelor
                                ;       E - ca parametru pentru rutina DELAY
                                ;       D - utilizat de rutina DELAY
                                ;
7895                            PRTCOL:
7895    D3 B0                           OUT     (PRTPORT),A     ; coloana primita se trimite la portul acelor
7897    1E 01                           LD      E,SETDEL        ; intirziere de ace active,
7899    CD 78A6                         CALL    DELAY           ; de 300 microsecunde
789C    3E 80                           LD      A,RESNEAD       ; comanda de ridicare a acelor
789E    D3 B0                           OUT     (PRTPORT),A     ; se trimite la port
78A0    1E 04                           LD      E,RESDEL        ; intirziere de ace resetate,
78A2    CD 78A6                         CALL    DELAY           ; de 1200 microsecunde
78A5    C9                              RET
                                ;
                                ;       Rutina DELAY
                                ;
                                ; Realizeaza o intirziere de ( 5,6 x BASEDEL x valoarea din E ) microsecunde.
                                ; Distruge: D,E
                                ; Registrii utilizati:
                                ;       E - parametrul de intirziere trimis de rutina apelanta
                                ;       D - numarator de intirziere
                                ;
78A6                            DELAY:
78A6    16 35                           LD      D,BASEDEL       ; valoarea de decrementat pentru intirzierea
78A8                            DEL:                            ; de baza ( 300 microsecunde )
78A8    15                              DEC     D               ; (4) se decrementeaza D
78A9    C2 78A8                         JP      NZ,DEL          ; (10) pina cind devine 0
78AC    1D                              DEC     E               ; (4) se decrementeaza E; daca nu s-a ajuns
78AD    C2 78A6                         JP      NZ,DELAY        ; (10) la 0, se revine la un nou ciclu a lui D
78B0    C9                              RET
                                        PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-66

                        ;
                        ;       Rutina MOTON
                        ;
                        ; Porneste motorul imprimantei.
                        ; Distruge: A
                        ; Registru utilizat:
                        ;       A - pentru transmiterea comenzii la port
                        ;
78B1                    MOTON:
78B1    3E 80                   LD      A,MOTSTART      ; comanda de pornire a motorului
78B3    D3 B0                   OUT     (PRTPORT),A     ; se trimite la port
78B5    C9                      RET
                        ;
                        ;       Rutina MOTOFF
                        ;
                        ; Asteapta ca semnalul de la limitatorul cursei capului de imprimare sa treaca
                        ; din 0 in 1,dupa care opreste motorul imprimantei.
                        ; Distruge: A
                        ; Registru utilizat:
                        ;       A - pentru citirea portului SYSIN si transmiterea comenzii de oprire.
                        ;
78B6                    MOTOFF:
78B6    DB 90                   IN      A,(SYSIN)       ; se citeste portul si
78B8    CB 7F                   BIT     CARLIMIT,A      ; se testeaza valoarea semnalului;cit timp
78BA    28 FA                   JR      Z,MOTOFF        ; este 0 ,se sta in bucla
78BC    3E 00                   LD      A,MOTSTOP       ; comanda de oprire a motorului
78BE    D3 B0                   OUT     (PRTPORT),A     ; se trimite la port
78C0    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-67


                        ;
                        ;
                        ;       Rutina SRCNBLK
                        ;
                        ; Cauta primul caracter diferit de spatiu in zona indicata de HL.Primeste de
                        ; la rutina apelanta adresa zonei in HL si numarul de octeti ai zonei in B.La
                        ; sfirsit HL indica adresa locatiei ce urmeaza caracterului gasit,B numarul
                        ; de locatii ramase,iar A contine caracterul gasit.
                        ; Distruge: A,B,H,L
                        ; Registrii utilizati:
                        ;       HL- pentru adresarea zonei
                        ;       B - numaratorul octetilor zonei
                        ;       A - pentru transmiterea caracterului rutinei apelante
                        ;
78C1                    SRCNBLK:
78C1    7E                      LD      A,(HL)          ; caracterul curent in A
78C2    23                      INC     HL
78C3    05                      DEC     B               ; se decrementeaza numaratorul
78C4    FE 20                   CP      BLANK           ; este spatiu ?
78C6    28 F9                   JR      Z,SRCNBLK       ; daca da,cautarea continua
78C8    C9                      RET
                                PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-68

                        ;
                        ;
                        ;       Rutina DROPOUT
                        ;
                        ; Este rutina de tratare a intreruperii nemascabile la caderea de tensiune.
                        ; Se salveaza toti registrii pe stiva,iar registrul indicator al virfului de
                        ; stiva in RAM,dupa care se intra in HALT.
                        ;
78C9                    DROPOUT:
78C9    F5                      PUSH    AF              ; se salveaza registrii
78CA    C5                      PUSH    BC
78CB    D5                      PUSH    DE
78CC    E5                      PUSH    HL
78CD    DD E5                   PUSH    IX
78CF    FD E5                   PUSH    IY
78D1    D9                      EXX
78D2    08                      EX      AF,AF
78D3    F5                      PUSH    AF
78D4    C5                      PUSH    BC
78D5    D5                      PUSH    DE
78D6    E5                      PUSH    HL
78D7    ED 73 6F28              LD      (STACKR),SP     ; se salveaza registrul SP
78DB    76                      HALT
                        ;       PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-69


                                ;
                                ;
                                ;       Rutina INT
                                ;
                                ; Deserveste cererile de intrerupere care sosesc cu o intermitenta de 2 ms.
                                ; Scopul ei este de a baleia dispozitivul de afisare ( imaginea afisata pe
                                ; LED-uri trebuie sa fie oglinda codurilor continute in buferul de afisaj
                                ; LEDBUF ).
                                ; REgistrii utilizati:
                                ;       B'- numarator circular cu valori intre 8 si 1
                                ;       C'- contine adresa portului de afisaj LEDPORT
                                ;       HL'-contine adresa curenta din LEDBUF
                                ;
78DC                            INT:
78DC    08                              EX      AF,AF           ; se salveaza registrul A
78DD    D9                              EXX                     ; si se comuta pe setul auxiliar de registrii
78DE    ED A3                           OUTI                    ; se transfera la LEDPORT valoarea de la adre-
                                                                ; sa curenta din LEDBUF si se decrementeaza
                                                                ; numaratorul circular
78E0    3A 6F27                         LD      A,(WITNESS)     ; se ia valoarea curenta a portului sistem din
78E3    E6 F8                           AND     MASK2           ; celula martor si se mascheaza,pentru a nu
                                                                ; afecta decit bitii implicati
78E5    B0                              OR      B               ; se pozitioneaza bitii de selectie
78E6    32 6F27                         LD      (WITNESS),A     ; si noua valoare se salveaza in celula martor
78E9    D3 90                           OUT     (SYSOUT),A      ; valoarea creata se trimite la port
78EB    AF                              XOR     A               ; A = 0
78EC    B0                              OR      B               ; se testeaza valoarea numaratorului circular
78ED    20 05                           JR      NZ,INT1         ; daca nu este 0,salt la sfirsit
78EF    21 6E31                         LD      HL,LEDBUF       ; daca este 0,se reinitializeaza variabilele
78F2    06 08                           LD      B,BUFLEN        ; implicate
78F4                            INT1:
78F4    D9                              EXX                     ; se revine la setul principal
78F5    08                              EX      AF,AF           ; se reface A
78F6    FB                              EI                      ; si se activeaza sistemul de intreruperi
78F7    ED 4D                           RETI
                                        PAGE
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-70


                                ;
                                ;       Tabela de texte MESSAGE
                                ;
                                ; Contine textele ce se tiparesc pe bonuri.Textele sint codificate in cod
                                ; intern.
                                ;
78F9                            MESSAGE:
78F9    81 08 20 1C                     DB      81H,08H,20H,1CH,18H,1CH         ; TOTAL:
78FD    18 1C
78FF    10 15 21                        DB      10H,15H,21H
7902    82 04 20 1C                     DB      82H,04H,20H,1CH,18H,1CH         ; TOTAL COD
7906    18 1C
7908    10 15 20 11                     DB      10H,15H,20H,11H,18H,12H
790C    18 12
790E    83 02 2D 1C                     DB      83H,02H,2DH,1CH,18H,1CH         ; TOTALUL ZILEI
7912    18 1C
7914    10 15 1D 15                     DB      10H,15H,1DH,15H,20H,1FH
7918    20 1F
791A    14 15 13 14                     DB      14H,15H,13H,14H
791E    84 01 20 1B                     DB      84H,01H,20H,1BH,14H,17H         ; SINTEZA
7922    14 17
7924    1C 13 1F 10                     DB      1CH,13H,1FH,10H
7928    85 01 20 20                     DB      85H,01H,20H,20H,20H,20H         ;    VINZARILOR
792C    20 20
792E    1E 14 17 1F                     DB      1EH,14H,17H,1FH,10H,1AH
7932    10 1A
7934    14 15 18 1A                     DB      14H,15H,18H,1AH
7938    86 0F                           DB      86H,0FH                         ; ( total )
793A    87 0F                           DB      87H,0FH                         ; ( rind gol )
793C    88 0F                           DB      88H,0FH                         ; ( nr.bon      casier )
793E    89 08 20 0D                     DB      89H,08H,20H,0DH,1EH,10H         ; *VA MULTUMIM*
7942    1E 10
7944    20 16 1D 15                     DB      20H,16H,1DH,15H,1CH,1DH
7948    1C 1D
794A    16 14 16 0D                     DB      16H,14H,16H,0DH
794E    8A 0F                           DB      8AH,0FH                         ; ( rind gol )
7950    8B 0F 20 1D                     DB      8BH,0FH,20H,1DH,17H,14H         ; UNIT. NR.
7954    17 14
```

```
MACRO-80 5.80   15-Jun-86       PAGE    1-71

7956    1C 0A 20 17             DB      1CH,0AH,20H,17H,1AH,0AH
795A    1A 0A
795C    8C 0F 20 20             DB      8CH,0FH,20H,20H,11H,10H         ;  CASA NR.
7960    11 10
7962    1B 10 20 17             DB      1BH,10H,20H,17H,1AH,0AH
7966    1A 0A
7968    8D 0F                   DB      8DH,0FH                         ; ( data )
796A    8E 0F                   DB      8EH,0FH                         ; ( rind gol )
                        ;
                        ;
                        ;       Generatorul de caractere pentru afisaj
                        ;
796C                    LEDGEN:
796C    C0                      DB      0C0H            ; 0
796D    F9                      DB      0F9H            ; 1
796E    A4                      DB      0A4H            ; 2
796F    B0                      DB      0B0H            ; 3
7970    99                      DB      99H             ; 4
7971    92                      DB      92H             ; 5
7972    82                      DB      82H             ; 6
7973    F8                      DB      0F8H            ; 7
7974    80                      DB      80H             ; 8
7975    90                      DB      90H             ; 9
                                PAGE
```

```
                        ;
                        ;
                        ; Generatorul de caractere pentru imprimanta
                        ;
7976                    PRTGEN:
7976 BE C5 C9 D1                DB      0BEH,0C5H,0C9H,0D1H,0BEH        ; 0 = 00H
797A BE
797B C0 C0 FF C2                DB      0C0H,0C0H,0FFH,0C2H,0C4H        ; 1 = 01H
797F C4
7980 C6 C9 D1 D1                DB      0C6H,0C9H,0D1H,0D1H,0E6H        ; 2 = 02H
7984 E6
7985 A2 D5 C9 C9                DB      0A2H,0D5H,0C9H,0C9H,0C1H        ; 3 = 03H
7989 C1
798A 90 FF 92 94                DB      90H,0FFH,92H,94H,98H            ; 4 = 04H
798E 98
798F B9 C5 C5 C5                DB      0B9H,0C5H,0C5H,0C5H,0A7H        ; 5 = 05H
7993 A7
7994 B1 CA CC C8                DB      0B1H,0CAH,0CCH,0C8H,0B0H        ; 6 = 06H
7998 B0
7999 83 85 89 F1                DB      83H,85H,89H,0F1H,81H            ; 7 = 07H
799D 81
799E B6 C9 C9 C9                DB      0B6H,0C9H,0C9H,0C9H,0B6H        ; 8 = 08H
79A2 B6
79A3 86 89 99 A9                DB      86H,89H,99H,0A9H,0C6H           ; 9 = 09H
79A7 C6
79A8 80 80 C0 80                DB      80H,80H,0C0H,80H,80H            ; . = 0AH
79AC 80
79AD                            DS      5                               ; FUNC ( nu se imprima )
79B2 88 88 BE 88                DB      88H,88H,0BEH,88H,88H            ; + = 0CH
79B6 88
79B7 94 88 BE 88                DB      94H,88H,0BEH,88H,94H            ; * = 0DH
79BB 94
79BC                            DS      5                               ; TOTAL ( nu se imprima )
79C1                            DS      5                               ; CLEAR ( nu se imprima )
79C6 FC 92 91 92                DB      0FCH,92H,91H,92H,0FCH           ; A = 10H
79CA FC
79CB A2 C1 C1 C1                DB      0A2H,0C1H,0C1H,0C1H,0BEH        ; C = 11H
79CF BE
```

```
                MACRO-80 5.80   15-Jun-86        PAGE    1-73

79D0    9C A2 C1 C1                     DB      9CH,0A2H,0C1H,0C1H,0FFH ; D = 12H
79D4    FF
79D5    C1 C9 C9 C9                     DB      0C1H,0C9H,0C9H,0C9H,0FFH ; E = 13H
79D9    FF
79DA    80 C1 FF C1                     DB      80H,0C1H,0FFH,0C1H,80H  ; I = 14H
79DE    80
79DF    C0 C0 C0 C0                     DB      0C0H,0C0H,0C0H,0C0H,0FFH ; L = 15H
79E3    FF
79E4    FF 82 8C 82                     DB      0FFH,82H,8CH,82H,0FFH   ; M = 16H
79E8    FF
79E9    FF B0 88 84                     DB      0FFH,0B0H,88H,84H,0FFH  ; N = 17H
79ED    FF
79EE    BE C1 C1 C1                     DB      0BEH,0C1H,0C1H,0C1H,0BEH ; O = 18H
79F2    BE
79F3    86 89 89 89                     DB      86H,89H,89H,89H,0FFH    ; P = 19H
79F7    FF
79F8    C6 A9 99 89                     DB      0C6H,0A9H,99H,89H,0FFH  ; R = 1AH
79FC    FF
79FD    B2 C9 C9 C9                     DB      0B2H,0C9H,0C9H,0C9H,0A6H ; S = 1BH
7A01    A6
7A02    81 81 FF 81                     DB      81H,81H,0FFH,81H,81H    ; T = 1CH
7A06    81
7A07    BF C0 C0 C0                     DB      0BFH,0C0H,0C0H,0C0H,0BFH ; U = 1DH
7A0B    BF
7A0C    9F A0 C0 A0                     DB      9FH,0A0H,0C0H,0A0H,9FH  ; V = 1EH
7A10    9F
7A11    C3 C5 C9 D1                     DB      0C3H,0C5H,0C9H,0D1H,0E1H ; Z = 1FH
7A15    E1
7A16    80 80 80 80                     DB      80H,80H,80H,80H,80H     ;   = 20H
7A1A    80
7A1B    80 80 A2 80                     DB      80H,80H,0A2H,80H,80H    ; : = 21H
7A1F    80
7A20    88 88 88 88                     DB      88H,88H,88H,88H,88H     ; - = 22H
7A24    88

                                ;
                                ;
                                ;
                                ;

                                END
```

```
                MACRO-80 5.80   15-Jun-86        PAGE    S

Macros:

Symbols:
ADDBCD  755D    ADDBYT  7569    ADDD    7436    ADDSOR  743A
AFTERC  72EC    ANULAR  0010    ASTEPT  779E    ASTER   000D
BASEDE  0035    BCDCI   7666    BCDCI1  7672    BCDLEN  0005
BEEP    77C3    BEEPBI  0007    BEGPR   735C    BLANK   0020
BLKDIS  00FF    BTESTT  7148    BTWCHD  0005    BTWTDE  0080
BUFLEN  0008    BUYMAS  0008    BUYTIC  7145    CARLIM  0007
CASE    7178    CASETA  7186    CHANGE  768C    CLBUFD  771F
CLDISP  7735    CLEAR   000F    CLEARC  7744    CLKEYB  7726
CLRKN   6EBD    CLRKNL  0003    CNTFUN  7194    CODDIG  7376
CODE    6E57    CODED   72E9    CODELE  72F5    COLDST  70A1
COMP    7811    COMPUT  7805    CONTR   7098    CONTRL  0009
CORREC  73D4    CTC     00C0    CTCCMD  0097    CTCCOU  00FA
CTCPRO  705F    DATBCD  6E08    DATE    6F0A    DATEL   0008
DAYBCD  6C00    DAYMAS  0002    DECPAR  7624    DEL     78A8
DELAY   78A6    DELAY0  7817    DELAY1  781A    DELAY2  781C
DESKN   6F01    DESKNL  0002    DISP1   76FD    DISPNR  719C
DISPNU  76B3    DISPSU  76F3    DIVIDO  7828    DIVID1  7832
DIVIDE  7821    DPNUM   76BB    DPPOIN  76CF    DROPOU  78C9
EBCD    6E0D    EBCDBC  7650    EBCDIN  75E7    EBCDLE  000A
EDBFRE  03E8    EDBTIM  0014    EDBUF   6E39    EDLINE  74EF
EDLLEN  0011    EMITBU  7458    EMITER  749F    ENDBUY  7157
ENDDP   76D4    ENDERA  7232    ENDERP  7357    ENDMOP  75E2
ENDNXP  736C    ENDOP   740E    ENDPRP  732B    ENDST   76AF
ENDSYN  72DE    ENDTDA  72BC    ENDTRA  720C    ENDTSO  7281
EPRERM  00AF    EPROMA  700A    EPROMB  7000    EPROME  7016
EPROML  1000    EPROMT  7003    ERACOR  7222    ERAEND  7243
ERAERR  7237    ERATIC  7219    ERPRIC  732C    ERRCYC  705B
ERRSYN  72D6    ERRTDA  72B0    ERRTSO  7275    ERSDFR  07D0
ERSDTM  0014    ETESTT  7223    EXPAND  774F    EXPCHR  775A
EXPLIN  7758    EXPSPA  7763    FCSDFR  0BB8    FCSDTI  0014
FILLBY  774A    FORMME  7640    FUNC    000B    FUNCNU  0007
GUEST8  6DF9    HEADPO  7089    IMPLCO  739B    INCODE  7370
INERTD  0070    INITSO  7082    INKEY   7771    INPRIC  72E3
INT     78DC    INT1    78F4    INTMOD  706B    INTPAR  760D
INTTAB  6FF8    INTVAR  7077    KEY1    0001    KEY2    0002
KEYBUF  6E29    KEYTAB  77B3    KYBDMA  000C    LAMPT   70CC
LEDBUF  6E31    LEDGEN  796C    LEDPOR  00A0    LMP     70D1
M10     75C8    MAINLO  712C    MASK1   000C    MASK2   00F8
MAXFUN  0002    MESSAG  78F9    MINUS   0022    MOTOFF  78B6
MOTON   78B1    MOTSTA  0080    MOTSTO  0000    MOVTMP  7705
```

```
                MACRO-80 5.80    15-Jun-86        PAGE    S-1

MUL10   75C7    MULT    75B0    MULTBC  758D    MULTEN  75B8
MULTOP  75CF    NEXTDE  762A    NEXTED  750C    NEXTKE  7302
NOKEY   77A9    NOPOIN  7208    NORMLI  7506    NORMSO  00BF
NREDLN  000E    NRFUNC  6F2A    NRORPT  000B    NSORTT  7532
NXPRIC  7358    NXTLIN  7784    ONEHZ   61A8    OPERFR  03E8
OPERRO  73FF    OPERTM  0014    OPFORB  73A8    OTASTA  7799
PACKAG  73C3    PACKBY  7658    PARAMP  71C6    PCHAR   786F
PCOL    788C    PLUS    000C    POINT   000A    PRCODD  737F
PRELPR  7643    PROCCL  7173    PROCDI  72FF    PROCFU  715B
PROCLO  7162    PRODUC  73B7    PRPRIC  7318    PRT18C  7869
PRTBUF  6E17    PRTCHA  787F    PRTCOL  7895    PRTGEN  7976
PRTLEN  0012    PRTLIN  783A    PRTPOR  00B0    PRTTIC  74E6
RAMBOT  6C00    RAMBYT  7021    RAMEME  00B7    RAMERH  2710
RAMERL  03E8    RAMERR  7031    RAMERT  00C8    RAMESD  7037
RAMINI  70E4    RAMLEN  0400    RAMRD   70BA    RAMT1   701B
RAMT2   70A4    RAMWR   70AA    RESDEL  0004    RESNEA  0080
SCLEN   0002    SENDFR  0320    SENDTM  0014    SETDEL  0001
SETN    71B8    SETUP   71B5    SETUPT  720D    SHIFTA  778D
SHIFTM  0003    SHIFTS  75A1    SHOPN   6EF0    SHOPNL  0003
SIMPBU  7137    SING    77E4    SING1   77E5    SORTAD  740F
SORTMA  0004    SORTTO  6C05    SPEC    7501    SRCNBL  78C1
STACK1  6FF8    STACK2  6FF0    STACKR  6F28    START   7000
STNUM   769C    STORDI  76D9    STORED  7638    STOREI  76E0
STOREN  7695    STPOIN  76AA    STRTTY  708F    SUBBCD  7575
SUBBYT  7581    SUBCOR  733E    SUBERR  7343    SUBSOR  7449
SUM     6E93    SYNTAN  75F2    SYNTH   728D    SYNTMA  0001
SYNTTO  7529    SYSIN   0090    SYSOUT  0090    TAKEBY  71FA
TCKNRL  0004    TCLOCK  0190    TDECEN  763E    TENDP   730F
TESTER  73D2    TESTFU  713E    TESTSU  733C    TICKNR  6EB3
TINTEN  7610    TMPBCD  6E03    TMPMOV  7712    TOTAL   000E
TOTDAY  7282    TOTSOR  7244    TRANLI  7519    TRANSP  71EB
TSHIFT  7794    WADROU  7051    WAITPO  7841    WARMST  7118
WITHOU  7634    WITHP   7630    WITNES  6F27    WORKBC  6DFE

No  Fatal error(s)
```

18 MEMENTO

Elaborarea unor materiale de sinteză, care să ajute reînțelegerea problemei de către aceeași persoană (peste ani de zile) sau de către altă persoană este o activitate care trebuie să însoțească elaborarea fiecărui pachet de software.

În acest capitol prezentăm următoarele materiale :

- **Harta memoriei RAM**
- **Fluxul de date (aproximația finală)**
- **Nivelele ierarhice ale software-ului realizat**
- **Lista rutinelor încorporate în produs**

Ne vom referi pe scurt la fiecare din ele.

18.1. Harta memoriei RAM

În memoria **RAM (1 kbyte)** se vor memora atît datele legate de vînzări, cît și variabilele implicite ale rutinelor elaborate precum și stiva de lucru a procesorului.

■ Rezervarea memoriei se face începînd de la adrese inferioare. Adresa inferioară a memoriei **RAM** considerată **($6C00_H$)** o numim **RAMBOT**.

- **DAYBCD (6C00)** — registrul care conține suma vînzărilor (deci total de bani din casă) îl vom dispune, datorită importanței lui, în primele celule RAM.
- Urmează totalurile de vînzări aferente celor 100 de clase de sortimente **($TOTSORT_0$—$TOTSORT_{99}$: 6C05—6DF8)**, fiecărei sume rezervîndu-i-se cîte 5 octeți. Această tabelă ocupă aproape jumătate din memoria RAM disponibilă.
- **GUESTBCD (6DF9)** este registrul **RAM** în care se generează suma clientului ;
- **WORKBCD (6DFE), TMPBCD (6E03)** și **DATBCD** (6E08) sînt regiștrii de lucru ai aritmeticii **BCD** cu virgulă fixă și fără semn ;
- **EBCD (6E0D)** — este o zonă de 10 octeți în care se vor depune numerele normalizate, în format **BCD extins.**

- **PRTBUF (6E17—6E28)** este bufferul de ieşire pentru imprimantă ;
- **KEYBUF (6E29—6E30)** şi **LEDBUF (6E31—6E38)** vor fi zonele de manevră pentru tastatură şi dispozitivul de afişaj.
- **EDBUF (6E39—6F26)** — este bufferul de editare pentru imprimantă în care pe lîngă cele 14 linii se disting următoarele puncte de intrare :

DATE : (6F0A) — 8 **byte** — zonă pentru memorarea datei calendaristice

DESKN : (6F01) — 2 **byte** — zonă pentru numărul casei de marcat

SHOPN : (6EF0) — 3 **byte** — zonă pentru numărul magazinului

CLRKN : (6EBD) — 3**byte** — zonă pentru identificatorul casierului

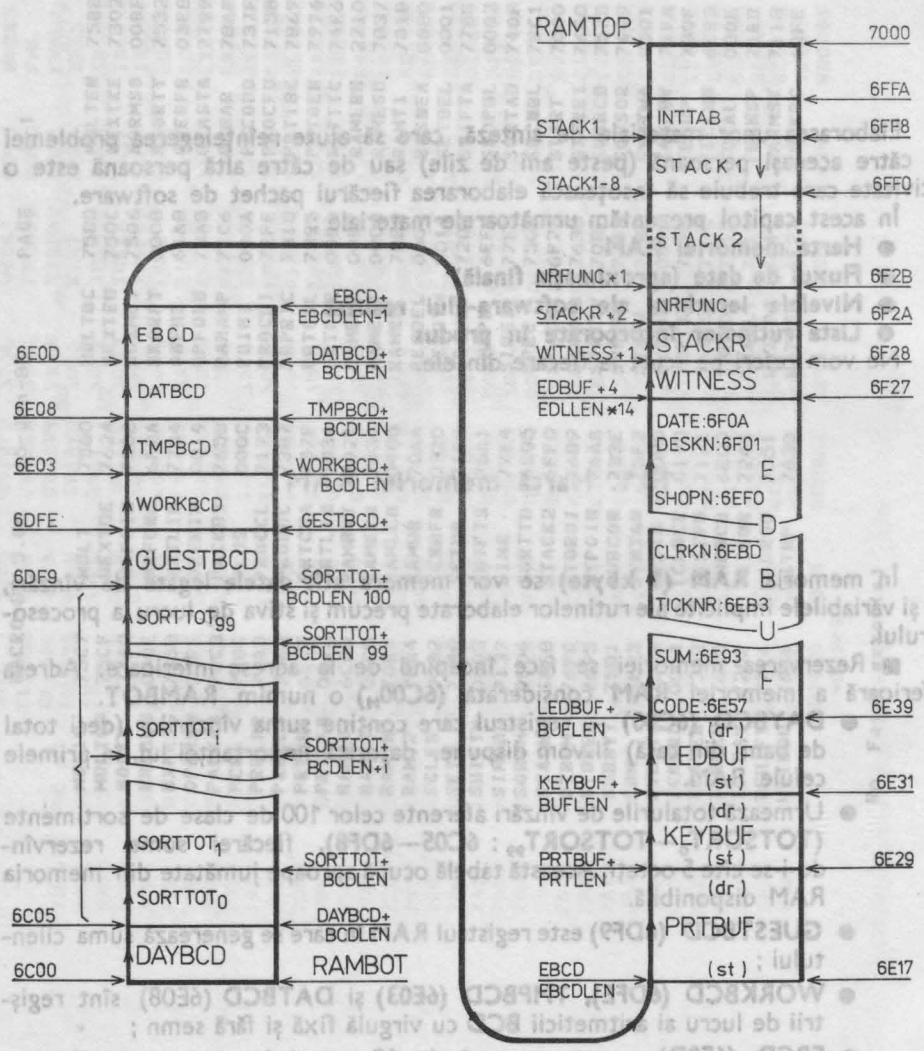


Fig. 18.1. Harta memoriei RAM (6C00—6FFF)

TICKNR : (6EB3) — 4 byte — zonă pentru numărul curent de bor
SUM : (6E93) — 11 byte — zonă pentru sumă totală
CODE : (6E57) — 2 byte — zonă pentru cod de sortiment

- **WITNESS (6F27)** este celula martor a portului de ieşire **SYSOUT**
- **STACKR (6F28 — 6F29)** — sînt locaţiile în care se salvează valoarea curentă a indicatorului de stivă, la apariţia căderii de tensiune **(DROPOUT)**
- **NRFUNC (6F2B)** — este o variabilă de lucru, care contorizează numărul de tastări **FUNC** (1 sau 2) în cazul bonurilor normale şi a celor de anulare ;

- **Elementele de stivă** se definesc faţă de capătul superior al memoriei **RAM (RAMTOP=6FFF)**, stiva fiind descrescătoare.
- Zona **(6FFF—6FF8)** este rezervată pentru **vectori de întrerupere**. Avînd o singură sursă de întreruperi (circuitul **CTC** pentru afişaj) vom avea un singur vector : **INTTAB (6FF8—6FF9)**.
- Zona **STACK1 (6FF8—6FF0)** este **stiva locală** care va fi folosită de către secvenţa program cuprinsă între adresa **0** (reset) şi punctul de ramificaţie **STARTTYPE**. Fiecare din cele două ramuri **CSTART** şi **WSTART** care pornesc din acest punct, îşi vor reiniţializa indicatorul de stivă : pornirea rece îl va iniţializa la **STACK2(6FF0)**, iar cea caldă după conţinutul celulei **STACKR**.
- Zona **STACK2(6FF0—6F2B)** este **stiva de lucru** propriu-zisă folosită de casa de marcat. Dimensiunea ei $C5_H$ va permite efectuarea a 62 de salvări succesive pe stivă, fără ca să pericliteze integritatea zonei de variabile a memoriei RAM.

18.2. Fluxul de date în casa de marcat (aproximaţia finală)

Schema din fig. 18.2. o completează pe cea din fig. 13.15 concretizînd numele unor activităţi şi suplimentînd figura cu ramuri care nu le-am întrezărit la elaborarea proiectului.

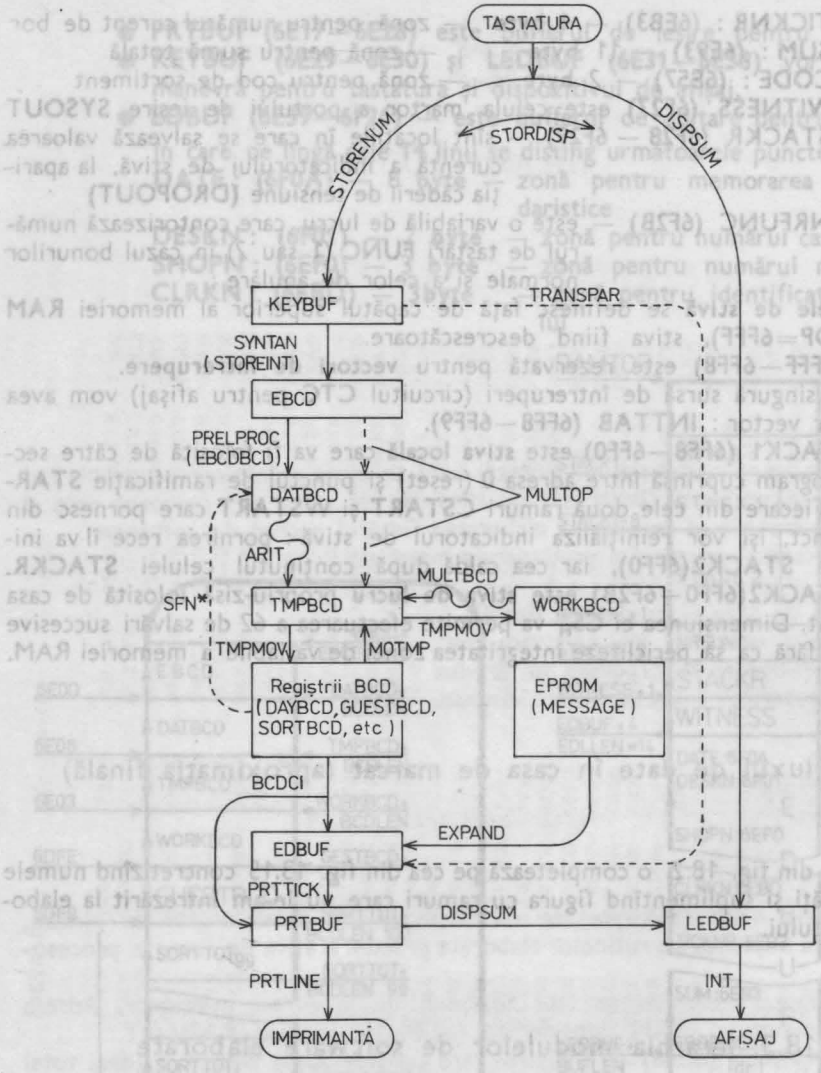
18.3. Ierarhia modulelor de software elaborate

Mulţimea rutinelor cuprinse în listingul din Cap. 17 a fost împărţită în trei clase **A, B, C.**

A — **software pur,** rezultat al specificaţiei funcţionale
B — secvenţă de **iniţializări şi teste** de trezire
C — **rutine de tratare fizică a perifericelor** (rutine „driver" sau „handler")

Cele trei clase fiind indispensabile, le-am acordat *aceeaşi prioritate*, neîncercînd o interclasificare a lor. De aceea în fig. 18.3. le regăsim ca ramuri paralele. Referindu-ne în continuare la modulele **ramurii A,** specificăm criteriile care au stat la baza ierarhizării :

- în **nivelul 0** am inclus modulele software care fac joncţiunea dintre specificaţia tehnică şi programele care vor realiza aceste funcţii impuse. Ele



SFN - Secvență fără nume

Fig. 18.2. Fluxul de date în casa de marcat (aproximația finală)

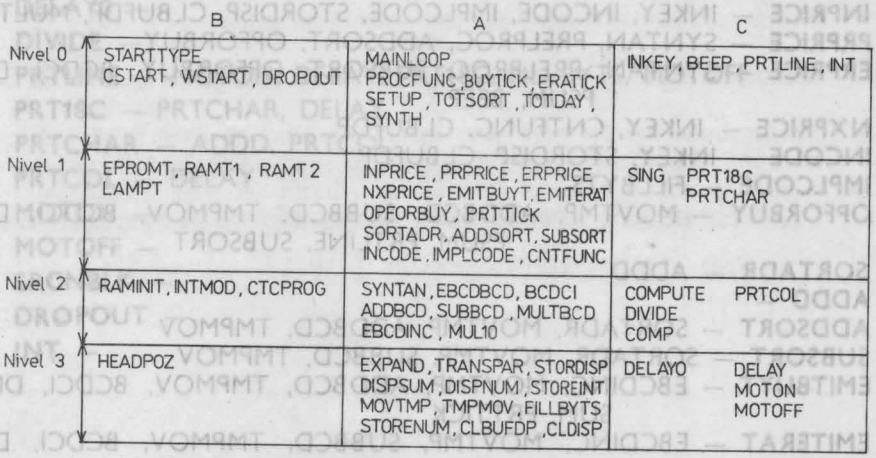


Fig. 18.3. Nivelele ierarhice ale software-ului realizat

creează de fapt cadrul pentru nivelele inferioare, fără să rezolve probleme concrete.

- în **nivelul 1** regăsim modulele care *rezolvă principial sarcinile impuse prin specificația tehnică.*
- în **nivelul 2** se găsesc *rutinele de uz general*, care sînt cvasiindependente de produs : *aritmetică BCD* și *programele de conversie.*
- ultimul nivel ierarhic **(nivelul 3)** cuprinde ***module slave*** de uz general, precum *și cîteva excepții* **(TRANSPAR,EXPAND)**

18.4. Lista rutinelor încorporate în produs

Ca un ultim element *dedicat propriei noastre* ***memorii*** elaborăm lista tuturor rutinelor din pachetul elaborat, specificînd în dreptul fiecăreia numele rutinelor pe care ea le apelează.

1. **START** (EPROMT, RAMT1, RAMT2, MOTON, MOTOFF, COLDSTRT, WSTART, LAMPT, RAMINIT)
2. **MAINLOOP** — INKEY, BUYTICK, PROCFUNC
3. **BUYTICK** — CLDISP, INPRICE, PRPRICE, NXPRICE, EMITBUYT
4. **PROCFUNC** — CLDISP, CNTFUNC, INKEY, DISPNUM, BUYTICK, SETUP, ERATICK, TOTSORT, TOTDAY, SYNTH
5. **CNTFUNC** — CLDISP, BEEP
6. **SETUP** — INKEY, CLBUFDP, STORDISP, TRANSPAR, DISPNUM, BEEP
7. **TRANSPAR** —
8. **ERATICK** — INPRICE, ERPRICE, PRPRICE, EMITERAT, BEEP, DISPNUM
9. **TOTSORT** — INCODE, SORTADR, BCDCI, DISPSUM, PRTTICK, BEEP, DISPNUM
10. **TOTDAY** — BCDCI, DISPSUM, PRTTICK, BEEP, DISPNUM.
11. **SYNTH** — PRTTICK, BEEP, DISPNUM

12. **INPRICE** — INKEY, INCODE, IMPLCODE, STORDISP, CLBUFDP, MULTOP
13. **PRPRICE** — SYNTAN, PRELPROC, ADDSORT, OPFORBUY
14. **ERPRICE** — SYNTAN, PRELPROC, SUBSORT, OPFORBUY, BCDCI, DISPSUM, BEEP
15. **NXPRICE** — INKEY, CNTFUNC, CLBUFDP
16. **INCODE** — INKEY, STORDISP CLBUFDP
17. **IMPLCODE** — FILLBYTS
18. **OPFORBUY** — MOVTMP, ADDBCD, SUBBCD, TMPMOV, BCDCI, DISPSUM, PRTLINE, SUBSORT
19. **SORTADR** — ADDD
20. **ADDD** —
21. **ADDSORT** — SORTADR, MOVTMP, ADDBCD, TMPMOV
22. **SUBSORT** — SORTADR, MOVTMP, SUBBCD, TMPMOV
23. **EMITBUYT** — EBCDINC, MOVTMP, ADDBCD, TMPMOV, BCDCI, DISPSUM, PRTTICK
24. **EMITERAT** — EBCDINC, MOVTMP, SUBBCD, TMPMOV, BCDCI, DISPSUM, PRTTICK, FILLBYTS
25. **PRTTICK** — SYNTTOTS, TRANLINE, PRTLINE, BEEP
26. **TRANLINE** —
27. **SYNTTOTS** — IMPLCODE, SORTADR, BCDCI, PRTLINE, EBCDINC
28. **ADDBCD** —
29. **SUBBCD** —
30. **MULTBCD** — TMPMOV, FILLBYTS, MUL10, ADDBCD
31. **MUL10** —
32. **MULTOP** — SYNTAN, EBCDBCD, MOVTMP, CLBUFDP
33. **EBCDINC** —
34. **SYNTAN** — FILLBYTS, SRCNBLK, STOREINT
35. **PRELPROC** — EBCDBCD MULTBCD
36. **EBCDBCD** —
37. **BCDCI** — MOVTMP
38. **STORENUM** —
39. **DISPNUM** —
40. **STORDISP** — STORENUM, DISPNUM
41. **STOREIND** —
42. **DISPSUM** — SRCNBLK, DISPNUM
43. **MOVTMP** —
44. **TMPMOV** —
45. **CLBUFDP** — CLKEYBUF, CLDISP
46. **CLKEYBUF** — FILLBYTS
47. **CLDISP** — FILLBYTS
48. **CLEARCOD** — CLBUFDP
49. **FILLBYTS** —
50. **EXPAND** —
51. **INKEY** —
52. **BEEP** — COMPUTE, DIVIDE, SING
53. **SING** — DELAY0
54. **COMPUTE** — DIVIDE, COMP
55. **COMP** —

56. **DELAY0** —
57. **DIVIDE** —
58. **PRTLINE** — MOTON, DELAY, PRT18C, FILLBYTS, MOTOFF
59. **PRT18C** — PRTCHAR, DELAY
60. **PRTCHAR** — ADDD, PRTCOL
61. **PRTCOL** — DELAY
62. **MOTON** —
63. **MOTOFF** —
64. **SRCNBLK** —
65. **DROPOUT** —
66. **INT** —

19

ÎN LOC DE PROBLEME ŞI EXERCIŢII
„Învierea" casei de marcat electronice pe calculatoarele PRAE şi aMIC

Cartea de faţă fiind dedicată în totalitate instruirii, problemele şi exerciţiile nu pot lipsi. Ne vom abate — încă o dată — de pe „drumurile bătute", enunţînd nu un şir de probleme, ci o temă de lucru. Finalizarea ei va oferi celor perseverenţi satisfacţia unei realizări palpabile.

19.1. Enunţul temei de lucru

După cum aţi realizat probabil, casa de marcat elaborată reprezintă o entitate de sine stătătoare, dotată cu interfeţe om-maşină proprii : tastatură, dispozitiv de afişaj cu LED-uri, imprimantă matricială.

Cu toate că proiectul software elaborat în capitolele precedente este complet şi este înregistrat în totalitate pe caseta **Visible—Z80**, *casa de marcat electronică nu va "învia"* pe calculatoarele dumneavoastră. *Motivul* este cît se poate de simplu : *rutinele de intrare/ieşire sînt specifice proiectului realizat* şi au fost elaborate pentru un hardware virtual, care diferă cert de cel al calculatoarelor **Prae** şi **aMIC**.

Puteţi vizualiza oricînd secvenţe ale acestui software, cu ajutorul simulatorului grafic încorporat în **Visible—Z80**, folosind comanda — **G** urmată de o **adresă** aleasă după plac din listingul capitolului 17. Dar astfel veţi putea vedea doar *„copacii"* nu şi *„pădurea"*. Dacă doriţi să vedeţi *casa de marcat „în acţiune"*, sau intenţionaţi să verificaţi exactitatea implementării, este necesar să modificaţi proiectul software realizat.

Iată deci enunţul temei de lucru :

Să se modifice software-ul, al cărui listing sursă se află în cap. 17, astfel încît el să funcţioneze pe calculatorul personal Prae sau aMIC.

Sarcina vi se va părea simplă sau mai complicată, în funcţie de nivelul dumneavoastră de cunoştinţe software şi în funcţie de informaţiile sistem pe care le posedaţi despre aceste calculatoare. Depinde de individualitatea fiecăruia în parte, dacă porneşte la drum sau nu. Cert este că *pentru începători aceasta este continuarea firească a procesului de asimilare*, în materie de programare. Să schiţăm pe scurt :

1. Incearcă să înţelegi programe existente, scrise de alţii.

2. *Incearcă să modifici programele înţelese, folosind criterii bine definite — exerciţii impuse.*
3. *Incearcă să defineşti criterii de modificare proprii şi implementează-le — figuri liber alese.*
4. *Solicită teme de lucru din partea unor programatori consacraţi, şi rezolvă-le.*
5. *Incearcă să enunţi teme de lucru şi implementează-le.*

Începătorii care au parcurs cartea pînă la acest punct, se află la cea de-a doua etapă din lista schiţată. Lor le vom elabora un ghid de lucru.

19.2. Ghid de lucru

19.2.1. Exerciţii impuse

■ **a.** Se va elabora o rutină, s-o numim **EMKEY (emulator de tastatură)**, menită să substituie rutina **INKEY**. Ea va citi tastatura calculatorului **(Prae** sau **aMIC)** şi va preda în registrul **A** codul tastei apăsate, neafectînd alţi regiştrii interni. Recomandăm folosirea *rutinelor de citire* a tastaturii *existente* în **EPROM**-urile calculatoarelor amintite.

● Reamintim faptul că ambele calculatoare folosesc coduri **ASCII**, motiv pentru care **EMKEY** *va trebui să efectueze şi o transcodare*, din cod **ASCII** în **codul intern**, specific casei de marcat (vezi cap. 13). Ca metodă de transcodare vă recomandăm utilizarea unei *tabele* a cărui *intrare* (adrese) o reprezintă codul **ASCII** şi a cărei *ieşire* (date) vor fi **codurile interne.**

● Ştiind că hardware-ul casei de marcat preconiza doar 16 taste, *recomandăm ca* **EMKEY** *să filtreze toate codurile de tastă care diferă de cele 16 definite* : **0, 1, 2, 3, 4, 5, 6, 7, 8, 9,.,+,** ∗, **F**-func, **C**-clear şi **T**-total. Astfel revenirea din **EMKEY** se va face doar în momentul obţinerii unui *cod intern valid.*

■ **b.** Se va elabora o rutină, fie ea **EMDP (emulator pentru dispozitivul de afişaj)**, care va substitui rutina **DISPNUM.**

● Definim pe ecran o *zonă pe care se vor afişa numerele* pe care casa de marcat le-ar afişa pe dispozitivul de afişaj cu **LED**-uri. Propunem ca zona respectivă să fie locată pe cel de-al doilea rînd (începînd din marginea inferioară a ecranului), centrat pe axa de simetrie verticală a imaginii.

● Recomandăm *utilizarea rutinelor de scriere pe ecran*, rezidente în EPROM-urile calculatoarelor.

Sarcinile pe care această rutină le va rezolva vor fi următoarele :

- *să afişeze* întocmai ca şi proiectul iniţial, *de la dreapta la stînga* ; în acest scop se vor folosi tehnicile de poziţionare a cursorului, specifice fiecărui calculator (vezi par. 19.4).
- va trebui *să efectueze o transcodare* din **cod intern** în cod **ASCII** utilizînd o metodă asemănătoare celei descrise la punctul **a.**
- înainte de a afişa un caracter (obligatoriu cifră sau punct zecimal), *rutina va deplasa conţinutul zonei ecran cu o poziţie la stînga*, cifra cea mai semnificativă ieşind din zona vizualizată (dacă numărul de cifre semnificative depăşeşte numărul de 8) — ea va trebui să dispară.

■ **c.** Se va elabora o rutină, a cărei nume să fie **SIMPRT (simulator pentru imprimantă)**, care va substitui rutina **PRTLINE.**

● Întrucît imprimanta lipseşte cu prisosinţă din majoritatea sistemelor realizate cu calculatoarele **Prae** şi **aMIC**, ne propunem să o simulăm tot pe ecranul "full-graphic" al acestor calculatoare personale.

● Definim pe ecran o *zonă* (fereastră) *de imprimare*. Rezervăm în acest scop cîte 18 caractere centrate pe axa verticală a imaginii, pe primele 24 rînduri (numărate din extrema superioară).

Funcţiile pe care **SIMPRT** le are de efectuat sînt analoage cu cele ale rutinei **PRTLINE**, pe care ea o va substitui. Să le recapitulăm :

- la apelare, ea *va transfera* conţinutul zonei tampon pentru imprimantă **(PRTBUF)** *pe rîndul inferior al zonei de imprimantă*, rezervată pe ecran (24 rînduri × 18 coloane) ;
- reţinem faptul că **PRTBUF** conţine informaţiile codificate în cod intern, iar rutinele de afişare pe ecran ale calculatoarelor **Prae** şi **aMIC** solicită coduri **ASCII** ; *se va efectua deci o transcodare* **cod intern → cod ASCII**, folosind tabela definită la punctul **b.** ; recomandăm să folosiţi şi în această tabelă un cod rezervat (ex. FF_H) pentru marcarea caracterelor nedefinite în proiectul de casă de marcat ; dacă rutina primeşte — printr-o întîmplare — un cod nepermis (FF_H), atunci ea îl va filtra, netrasmiţîndul spre afişare ;
- înainte de a „imprima" un rînd (pe rîndul inferior al zonei ecran de 24 × 18), *conţinutul zonei de imprimantă se va muta în sus (defilare) cu un rînd*, şi se *va şterge* (afişînd 18 blancuri) *rîndul inferior al zonei*, rînd pe care urmează să se facă imprimarea ;
- după afişarea rîndului dorit, **SIMPRT** *va şterge conţinutul zonei* **PRTBUF**, aidoma rutinei **PRTLINE.**

■ **d.** Se va elabora o rutină **EMCLDP (emularea ştergerii dispozitivului de afişaj)**, care va substitui rutina **CLDISP.**

● Întrucît rutina **CLDISP** „şterge" dispozitivl de afişaj în mod nemijlocit, apelînd rutina **FILLBYTS**, **EMCLDP** *va trebui să „umple" cu blancuri (spaţii) zona ecran* care este menită să înlocuiască dispozitivul de afişaj cu **LED**-uri.

■ **e.** Se va modifica rutina CNTFUNC, astfel încît ea să afişeze valoarea **numărătorului de tastări succesive FUNC** *pe ecran*, în extremitatea stîngă a zonei de afişaj (definită la punctul b.)

● Recomandăm utilizarea rutinelor de scriere pe ecran, rezidente, precum şi folosirea tehnicilor de adresare a cursorului.

■ **f.** *Se va elimina rutina* **BEEP**, pentru a evita îngreunarea implementării modificărilor. (Rutina **BEEP** acţionează portul de ieşire **SYSOUT**. Emiţînd valori neadecvate pe porturile sistem ale calculatoarelor **Prae** şi **aMIC**, se pot întîmpla lucruri absolut imprevizibile). Implementarea acestei rutine va fi făcută de cei mai ambiţioşi, doar în momentul în care ei vor cunoaşte bine structura hardware a calculatorului pe care lucrează. *Eliminarea rutinei* **BEEP** *se face substituind primul octet al rutinei cu codul instrucţiunii* **RET** ($C9_H$).

■ **g.** *Se analizează secvenţa de trezire* a casei de marcat, *eliminînd* (prin substituire cu NOP-uri 00_H) *toate instrucţiunile sau secvenţele care ar putea deranja funcţionarea calculatorului* (ex. testele de trezire sînt inutile, fiincă calculatorul este oricum trezit în momentul încărcării programului **Visible—Z80** ; se elimină referinţele la sistemul de întreruperi **EI,IM x** ; se elimină iniţializarea portului **SYSOUT** ; etc.).

■ Vom *considera remanierile drept bune dacă* comanda **C7000,712C(CR)** lansată din monitorul rezident al lui **Visible—Z80** *se execută fără cusur*, făcîndu-se revenirea în monitor odată cu atingerea adresei de breakpoint (**$712C_H$**). Amintim faptul că nu se vor elimina secvențele de inițializare ale variabilelor definite în proiectul de casă de marcat.

■ Programele nou elaborate se vor asambla în spațiul de adrese **6800 —$6DFF_H$**.

■ În *corpul* inițial al *programului* de casă de marcat cuprins în spațiul de adrese **7000 —$7A24_H$** *se vor efectua substituirile necesare, pentru a devia salturile* : din rutinele înlocuite, la rutinele înlocuitoare. În fiecare rutină înlocuită se vor modifica primii trei octeți, care vor fi înlocuiți cu cîte o instrucțiune de salt necondiționat la începutul rutinei înlocuitoare.

■ Vă recomandăm să *salvați programul modificat*, pe casetă magnetică, *sub forma unei înregistrări de sine stătătoare*, înregistrînd conținutul zonei de memorie : **6800_H—$7FFF_H$**.

Dacă rutinele elaborate și modificările făcute sînt corecte, atunci la comanda **C7000(CR)**, pe ecranul calculatorului dumneavoastră va „învia" casa de marcat electronică, proiectată în partea a III-a a cărții de față. Răbdare și perseverență.

Ajunși la sfîrșitul noului proiect, constatăm că software-ul inițial al casei de marcat a trecut destul de bine „proba de foc" a modificărilor, cerință considerată drept esențială (vezi experiențele formulate în cap. 10).

Trebuie totuși să recunoaștem că s-ar fi putut și mai bine. Într-un caz ideal, în corpul programului ar fi trebuit să facă doar cinci modificări (INKEY, DISPNUM, PRTLINE, BEEP, INIT) față de cele șapte enunțate mai sus.

Asta e ! ce să-i facem ? N-am rezistat ispitei de a trata dispozitivul de afișaj în mod neconvențional, renunțînd la serviciile rutinei DISPNUM în două cazuri : CLDISP, CNTFUNC. Atenție, nu e bine să alegi calea cea mai ușoară !

19.2.2. Figuri „liber alese"

Cei care au elaborat lucrarea enunțată la paragraful 19.1. și definită la paragraful 19.2.1. vor putea să treacă mai departe dînd frîu liber imaginației lor. aducînd noi modificări proiectului de casă de marcat. Nu vrem să vă influențăm, exercițiile fiind „liber alese", dar amintim cîteva posibilități intuite de noi :

■ dispozitivul de afișaj emulat pe ecran, să afișeze caractere similare cu cele pe care le-ar vizualiza afișorul cu **LED**-uri (cifre formate din 7 segmente — și nu caracterele proprii ale calculatoarelor **Prae** și **aMIC**) ;

■ în zona de imprimare rezervată pe ecran, să apară nu caracterele proprii ale calculatorului, ci cele definite în **PRTGEN** ;

■ „imprimarea" pe ecran să fie asortată de un zgomot specific imprimantelor matriciale cu ace, zgomot realizat pe cale software ; etc. ....

Cu cît proiectul rezultant va aproxima mai bine casa de marcat virtuală, definită în capitolul 11., cu atît emularea va fi mai perfectă.

Cei care au trecut prin etapele definite la paragraful 19.2. pot acum continua procesul de perfecționare în domeniul programării, continuînd de la punctul 4 al îndrumarului din paragraful 19.

Pe paginile următoare redăm lista codurilor ASCII, și cea a rutinelor uzuale ale calculatoarelor Prae și aMIC, venind astfel în ajutorul celor care își propun elaborarea lucrării enunțate în paragraful 19.1.

19.3. Tabela codurilor ASCII

Tab. 19.1. Codurile ASCII

CODURILE ASCII

HEX	MSD	0	1	2	3	4	5	6	7
LSD	BITS	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SPACE	0	@	P	-	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O		o	DEL

NUL — Null
SOH — Start of Heading
STX — Start of Text
ETX — End of Text
EOT — End of Transmission
ENQ — Enquiry
ACK — Acknowledge
BEL — Bell
BS — Backspace
HT — Horizontal Tabulation
LF — Line Feed
VT — Vertical Tabulation
FF — Form Feed
CR — Carriage Return
SO — Shift Out
SI — Shift In
DLE — Data Link Escape
DC — Device Control
NAK — Negative Acknowledge
SYN — Synchronous idle
ETB — End of Transmission Block
CAN — Cancel
EM — End of Medium
SUB — Substitute
ESC — Escape
FS — File Separator
GS — Group Separator
RS — Record Separator
US — Unit Separator
SP — Space (Blank)
DEL — Delete

MSD reprezintă cifra cea mai semnificativă (Most Significant Digit — D6—D4) iar LSD cifra cea mai puţin semnificativă (D3—D0) a codului ASCII.

19.4. Lista rutinelor uzuale ale microcalculatoarelor PRAE și aMIC

În cei cîte 16 kocteți de memorie fixă a calculatoarelor **PRAE** și **aMIC** rezidă cîte un **program monitor** și cîte un **interpretor BASIC.** Aceste pachete software conțin multe rutine care pot fi folosite de către utilizatori, prin apelarea lor din programe proprii, scrise în limbaj de asamblare.

Redăm lista rutinelor importante :

PRAE

CI — **Consol Input.** Citește un caracter de la consola selectată (tastatura proprie sau interfața serială). Codul caracterului citit se găsește în registrul **A.**
3151_H

CO — **Consol Output.** Transmite caracterul al cărui cod se află în registrul **C** la consola selectată.
3154_H

CSTS — **Consol Status.** Verifică starea consolei de intrare selectate și remite în **A** — FF_H dacă nu s-a apăsat nici o tastă, sau codul caracterului tastat.
3169_H

SI — **Serial Input.** Citește un caracter de pe interfața serială. Codul caracterului citit se găsește în registrul **A.**
$317B_H$

SO — **Serial Output.** Transmite caracterul al cărui cod se află în registrul **C,** la interfața serială.
$316C_H$

KYI — **Keyboard Input.** Citește un caracter de la tastatura proprie. Codul caracterului citit se găsește în registrul **A.**
$318D_H$

CRTO — **CRT Output.** Afișează pe ecran pe poziția indicată de variabila **PNT** caracterul al cărui cod se află în registrul **C.** Avansează cursorul cu o poziție la dreapta.
3190_H

KSTS — **Keyboard Status.** Rutina are acțiune similară cu **CSTS,** dar se referă doar la tastatura proprie.
$31A2_H$

SSTS — **Serial Status.** Rutină similară cu **KSTS,** dar se referă la interfața serială.
$319F_H$

CASOUT — Înregistrează pe caseta magnetică un bloc de date aflat în bufferul de casetă 4080_H—$40FF_H$.
3187_H

CASIN — Citește de pe casetă magnetică un bloc de date și îl depune în bufferul de casetă 4080_H—$40FF_H$.
$318A_H$

RAMT — Test nedistructiv pentru memoria **RAM.** Afișează adresa primei celule din care nu s-a putut citi corect valoarea înscrisă.
$31AE_H$

EPROMT — Test pentru memoria **EPROM.** Afișează un mesaj de eroare și numărul **EPROM**-ului (0, 7) găsit a fi eronat.
$31A8_H$

CRLF — Transmite la consola de ieșire selectată, caracterele **CR** ($0D_H$) și **LF** ($0A_H$) (Retur de car și avans de linie).
$01F2_H$

BLK — Transmite la consola de ieșire selectată, **caracterul spațiu** (20_H).
$01E5_H$

LBYTE — Listează la consola de ieșire selectată, **valoarea hexazecimală a octetului** cuprins în registrul **A.**
$024C_H$

LADR — Listează la consola de ieșire selectată, **valoarea hexazecimală a celor doi octeți** aflați în registrul dublu **HL.**
0247_H

Variabile sistem importante.

BEG : 4004_H—4005_H. Conține adresa de început ecran pentru rutinele de tipărire ($E000_H$ la **PRAE 48K** și 6000_H la **PRAE 16K**).

BEGPLT : 401A—401 B_H. Conține adresa de început ecran pentru rutinele de grafică.

FIN : 4008—4009$_H$. Conține adresa primului octet după sfîrșitul memoriei de imagine (**0000$_H$** la **PRAE 48K** și **8000** la **PRAE 16K**).

PNT : 4006—4007$_H$. Adresa curentă a cursorului de pe ecran. Utilizatorul îi poate acorda valori în gama **E000$_H$—FF00$_H$** la **PRAE 48K** și **6000 —7F00$_H$** la **PRAE 16K**.

UPAD : 4010$_H$ Parametrii interfeței seriale.

+0 Martorul portului de ieșire sistem

+1 Adresa portului de ieșire (80$_H$).

+2 Număr de biți utili ai cuvîntului transferat.

+3 Viteza de transfer : FC$_H$—300 baud, 7E$_H$—600 baud, etc.

+4 Tipul de paritate : $b_0 = 0$ fără paritate ; $b_0 = 1$ și $b_1 = 0$ paritate pară ; $b_0 = 1$ și $b_1 = 1$ paritate impară.

+5 Numărul biților de stop (1 sau 2).

DENSIT : 4018$_H$ Viteza de înregistrare a datelor pe caseta magnetică. Poate varia în limite largi 6—255, viteza de transfer fiind invers proporțională cu acest număr. La trezire se inițializează la **0A$_H$** (2400 bit/s).

aMIC

a) Pentru calculatoarele echipate cu monitorul V0.1.

CIN 07FD$_H$ — **Consol Input.** Citește un caracter de la tastatură și îl furnizează în **A.**

COUT 0FEA$_H$ — **Consol Output.** Trimite la display caracterul conținut de registrul C (codul **ASCII** și îl afișează în poziția curentă a cursorului de pe ecran).

KIN 07F7$_H$ — **Cassette Input.** Citește de pe casetă un fișier în memoria microcalculatorului, la adresa de la care a fost salvat.

KOUT 07F4$_H$ — **Cassette Output.** Înscrie pe casetă un fișier din memoria calculatorului. La apelare **HL** va conține adresa de început, iar DE lungimea zonei de salvat.

Variabile sistem importante.

6000$_H$: **Numărul rîndului** alfanumeric, în care se află poziționat cursorul pe ecran. Va fi cuprins în limitele 00—1F$_H$, 00 corespunzînd primului rînd de pe ecran.

6001$_H$: **Numărul coloanei** alfanumerice, în care se află poziționat cursorul pe ecran. Valoarea va fi cuprinsă în limitele 00—1D$_H$, 00 corespunzînd primei coloane, din stînga ecranului.

b) Pentru calculatoarele echipate cu Monitor Z80 V0.0

Față de cele de la **pct. a)** apar diferențe de adresă la rutinele de casetă :

KIN — 3C1C$_H$

KOUT — 3BAE$_H$ — parametrii de apel : **HL** — adresă de început
BC — număr total de octeți
DE — număr fișier (4 cifre hexa).

c) Pentru calculatoarele echipate cu MON. aMIC V0.3 sau monitorul DEST.

Toate rutinele se apelează cu **CALL 0005$_H$.**

Octetul conținut în registrul **C** va defini **funcția** de efectuat, iar **D și E conțin eventualii parametri.**

C=0 : **RESET.** Ecranul este șters, variabilele cuprinse în zona 6000_H — —$60FF_H$ sînt inițializate cu 0.

C=1 : **CONIN.** Citește caracter de la consolă în registrul **A**.

C=2 : **CONOUT.** Scrie caracterul din registrul E la consolă.

C=3 : **RIN.** Citește caracterul de pe interfața serială (în **A**).

C=4 : **POUT.** Scrie caracter la interfața serială (din **E**).

C=5 : **LOUT.** Imprimă caracter pe miniimprimantă (din **E**).

C=9 : **WSTRIN.** Scrie șir de caractere la consolă. (**DE** adresă șir de caractere, care se termină cu „**$**" sau 00_H).

$C=0A_H$: **RSTRIN.** Citește și editează buffer consolă (**DE** adresă buffer).

$C=0B_H$: **CSTS.** Citește starea consolei. (Dacă s-a tastat caracter atunci $A=FF_H$, dacă nu atunci $A=00_H$).

Bibliografie

1. LUPU, CR., STĂNESCU, ST. Microprocesoare. Circuite. Proiectare, Editura Militară, București, 1986
2. LUPU, CR., etc. Microprocesoare. Aplicații. Editura Militară, București, 1982.
3. KRIZSÁN, GYÖRGY, A ZILOG cég mikroprocesszor családjai — vol. I — LSI OMIKK Budapest, 1984
4. BALTAC, VASILE Optimizarea sistemelor de operare ale calculatoarelor numerice, Editura Facla, Timișoara, 1974
5. MUNTEANU, EMIL, COSTEA V., MITROV, M. Programarea în limbaje de asamblare ASSIRIS, Editura Tehnică, București, 1976
6. MUNTEANU, EMIL Utilizarea calculatoarelor în prelucrarea datelor. Sistemul de operare, vol. I., Editura Dacia, Cluj-Napoca, 1974
7. BALTAC, V., ROMAN D. Calculatoarele electronice, grafica interactivă și prelucrarea imaginilor, Editura Tehnică, București, 1985
8. CĂPĂȚÎNĂ DAN, etc. Proiectarea cu microprocesoare, Editura Dacia, Cluj-Napoca, 1983
9. MUREȘAN, T., etc. Microprocesorul 8080 în aplicații, Editura Facla Timișoara, 1981
10. TOACȘE, GH. Introducere în microprocesoare, Editura Științifică și Enciclopedică, București, 1985.
11. PETERSON, J.L. Computer Organization and Assembler Language Programming, Academic Press, New York, 1978.
12. LESEA A. — ZAKS, R. Technique d'interface aux microprocesseurs, Sybex Inc. Paris, 1979
13. ZAKS, R. Programming the Z80, Sybex. Inc. Berkeley, 1982
14. WILLIAMS, S. Programming the 68000, Sybex, Inc. Berkeley, 1985
15. COFFRON, J. W. Programming the 8086, 8088, Sybex. Inc. Berkeley, 1983
16. BROOKS, F. P. The Mythichal Man Month., Addison—Wesley Reading (Mass.), 1979
17. DIJKSTRAA, E. W. A Discipline of Programming. Prentice Hall, New Jersey, 1976
18. WEINBERG, G. M. The Psychology of Computer Programming, Van Nostrand Reinhold Co, New York, 1975
19. YOURDON, E. N. Techniques of Program Structure and Design, Prentice Hall New Jersey, 1975
20. YOURDON, E. N. Managing the Structured Techniques, Yourdon Press, New York, 1979
21. KNUTH, D. E. The Art of Computer Programming. vol. I — Fundamental Algorithms. Addison—Wesley Reading (Mass.), 1973
22. KERNIGHAN, B. — PLANGER, P. Software Tools. Addison—Wesley Reading (Mass.), 1976
23. PATRUBÁNY, M. Mikroprocesszorok, rev. Korunk Füzetek nr. 2, Cluj-Napoca, 1983
24. KNUTH D. E. Tratat de programare a calculatoarelor I—III, Editura Tehnică, București, ed. 1978—85.
25. GOLDSTINE, H. H. Neumann szerepe a számítástechnikában (in Neumann János élete és munkássága), Budapest, 1979
26. PETRESCU, A. etc. Totul despre... calculatorul personal aMIC. Editura Tehnică București, 1985
27. JENSEN, K. — WIRTH, N. Pascal, User Manual and Report. Lecture Notes in Computer Science nr. 18, Springer Verlag, Berlin, 1976.
28. CIOCÎRLIE, H., ELES P., ș.a. Limbajele de programare Pascal și Pascal concurent. Editura Facla Timișoara, 1985
29. MAKARA ERNÖNÉ Tervezési segédlet a Z80 (MK3880) tipusú μP alkalmazásához. I rész Ipari Informatikai Központ Budapest. 1982
30. *** Z80. Technical Manual. Mostek, 1979
31. *** Data Catalog. SGS ATES, 1982

1— 2. L. Dumitrașcu	**Învățăm microelectronică interactivă conversînd în BASIC. Totul despre... BASIC în 14 conversații și 7 sinteze pe Felix C, CORAL, INDEPENDENT, Felix PC, M 118, TPD, HC. 85, aMIC, PRAE, COMMODORE, AMSTRAD și compatibile,** vol. 1 și 2.
3— 4. A. Petrescu și colectiv IPB, ITCI, Fabrica de calculatoare, Liceul Dimitrie Cantemir, CNOP	**ABC de calcul electronic. Totul despre... HC 85,** vol. 1, și vol. 2, **(o parte a tirajului cu 2 casete cu programe, acționînd calculatoare personale HC 85 și compatibile SINCLAIR SPECTRUM)**
5— 6. A. Tănăsescu și colectiv IPB, ISPIF	**Grafică asistată de calculator: Programe Fortran pe minicalculatoare, pentru reprezentări geometrice,** vol. 1 și vol. 2.
7—12. Colective largi	**Automatică, management, calculatoare (AMC). Serie continuă de instruire, informare, sinteze, cercetări aplicative în sisteme electronice, automate, informatice, de conducere,** volumele 56—51. **Echipamente electronice și tehnică de calcul — manuale de utilizare. Calculatoare personale, programe ș.a.**
13. A. Davidoviciu și colectiv ITCI, ASE	**Sistemul de operare MIX și limbajul MACRO pentru minicalculatoare CORAL/INDEPENDENT,** 2 volume.
14—15. I. Văduva, V. Baltac, Florescu V și colectiv ASE, ITCI	**Informatică economică,** 2 volume
16. Rusu O., Brudaru I.	**Echilibrarea liniilor flexibile.**
17. P. Constantinescu	**Sinergia, informația și geneza sistemelor.**

Se difuzează prin unitățile centrelor de librării, spre care se îndrumă întreprinderile și cititorii.

PENTRU ACESTE CARȚI SE POT FACE, TOTUȘI, ȘI COMENZI FERME LA EDITURA TEHNICĂ, PIAȚA SCÎNTEII 1, BUCUREȘTI.

Comenzile întreprinderilor se semnează de director și contabil șef, cele ale cititorilor individuali au indicată adresa exactă. Comenzile se trimit de editură la centrele de librării, cu indicarea unor priorități de satisfacere a lor. Plata nu se face decît la primirea exemplarelor.

• In prima parte a volumului 1 se prezintă exhaustiv structura și funcționarea amănunțită a microprocesorului, iar în partea a doua sînt grupate: manualul de utilizare a casetei, comparația între limbajele de asamblare ale celor două microprocesoare uzuale (Z 80 și I 8080), fișele detaliate ale tuturor instrucțiunilor microprocesorului Z 80, pe clase și grupe, lista instrucțiunilor publicate de firmă și a celor descoperite de utilizatori ș.a.

• Volumul 2 este constituit din proiectarea completă a unei case de marcat electronice cu microprocesor Z 80, pornind de la principiile și metodele proiectării-programării structurate în limbaj de asamblare, pînă la listing-ul amplu comentat al tuturor modulelor proiectului.

• Ultimul capitol este „o provocare" și totodată un test: cititorii cărții ghidați de indicațiile autorului sînt invitați să „invie" casa de marcat pe ecranul televizorului.

• În esență, o carte originală, de largă respirație, scrisă de un reputat specialist, o ediție complexă (structură, casetă, culoare ș.a.), care deschide în Biblioteca de Automatică, Informatică, Electronică, Management (BAIEM), ciclul de produse-program, ce va continua curînd.

ISBN 973-31-0009-9
ISBN 973-31-0007-2

Vol. I–II